

**MOTOROLA**

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

8-BIT MICROPROCESSING UNIT

The MC6809E is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the M6800 Family has major architectural improvements which include additional registers, instructions, and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The MC6809E has the most complete set of addressing modes available on any 8-bit microprocessor today.

The MC6809E has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications. External clock inputs are provided to allow synchronization with peripherals, systems, or other MPUs.

MC6800 COMPATIBLE

- Hardware — Interfaces with All M6800 Peripherals
- Software — Upward Source Code Compatible Instruction Set and Addressing Modes

ARCHITECTURAL FEATURES

- Two 16-Bit Index Registers
- Two 16-Bit Indexable Stack Pointers
- Two 8-Bit Accumulators can be Concatenated to Form One 16-Bit Accumulator
- Direct Page Register Allows Direct Addressing Throughout Memory

HARDWARE FEATURES

- External Clock Inputs, E and Q, Allow Synchronization
- TSC Input Controls Internal Bus Buffers
- LIC Indicates Opcode Fetch
- AVMA Allows Efficient Use of Common Resources in a Multiprocessor System
- BUSY is a Status Line for Multiprocessing
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- Interrupt Acknowledge Output Allows Vectoring By Devices
- Sync Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle RESET
- Single 5-Volt Supply Operation
- NMI Inhibited After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write Data for Dynamic Memories

SOFTWARE FEATURES

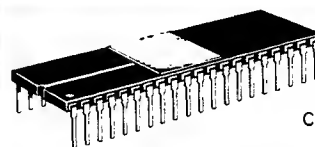
- 10 Addressing Modes
 - M6800 Upward Compatible Addressing Modes
 - Direct Addressing Anywhere in Memory Map
 - Long Relative Branches
 - Program Counter Relative
 - True Indirect Addressing
 - Expanded Indexed Addressing
 - 0-, 5-, 8-, or 16-Bit Constant Offsets
 - 8- or 16-Bit Accumulator Offsets
 - Auto-Increment/Decrement by 1 or 2
- Improved Stack Manipulation
- 1464 Instruction with Unique Addressing Modes
- 8 × 8 Unsigned Multiply
- 16-Bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

MC6809E

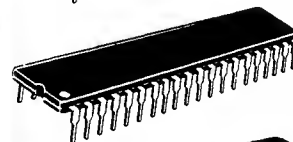
HMOS

(HIGH-DENSITY N-CHANNEL, SILICON-GATE)

8-BIT MICROPROCESSING UNIT



L SUFFIX
CERAMIC PACKAGE
CASE 715



P SUFFIX
PLASTIC PACKAGE
CASE 711



S SUFFIX
CERDIP PACKAGE
CASE 734

PIN ASSIGNMENT

| | | | |
|------|----|----|-------|
| VSS | 1 | 40 | HALT |
| NMI | 2 | 39 | TSC |
| TRO | 3 | 38 | LIC |
| FIRO | 4 | 37 | RESET |
| BS | 5 | 36 | AVMA |
| BA | 6 | 35 | Q |
| VCC | 7 | 34 | E |
| A0 | 8 | 33 | BUSY |
| A1 | 9 | 32 | R/W |
| A2 | 10 | 31 | D0 |
| A3 | 11 | 30 | D1 |
| A4 | 12 | 29 | D2 |
| A5 | 13 | 28 | D3 |
| A6 | 14 | 27 | D4 |
| A7 | 15 | 26 | D5 |
| A8 | 16 | 25 | D6 |
| A9 | 17 | 24 | D7 |
| A10 | 18 | 23 | A15 |
| A11 | 19 | 22 | A14 |
| A12 | 20 | 21 | A13 |

MAXIMUM RATINGS

| Rating | Symbol | Value | Unit |
|--|-----------|--|------|
| Supply Voltage | V_{CC} | -0.3 to +7.0 | V |
| Input Voltage | V_{in} | -0.3 to +7.0 | V |
| Operating Temperature Range MC6809E, MC68A09E, MC68B09E MC6809EC, MC68A09EC, MC68B09EC | T_A | T_L to T_H 0 to +70 -40 to +85 | °C |
| Storage Temperature Range | T_{stg} | -55 to +150 | °C |

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

| Characteristic | Symbol | Value | Unit |
|--|---------------|-----------------|------|
| Thermal Resistance Ceramic Cerdip Plastic | θ_{JA} | 50 60 100 | °C/W |

POWER CONSIDERATIONS

The average chip-junction temperature, T_J , in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

Where:

T_A = Ambient Temperature, °C

θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{PORT}$

$P_{INT} = I_{CC} \times V_{CC}$, Watts — Chip Internal Power

P_{PORT} = Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K + (T_J + 273^\circ\text{C}) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D \cdot (T_A + 273^\circ\text{C}) + \theta_{JA} \cdot P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A . Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A .

DC ELECTRICAL CHARACTERISTICS ($V_{CC} = 5.0 \text{ V} \pm 5\%$, $V_{SS} = 0 \text{ Vdc}$, $T_A = T_L$ to T_H unless otherwise noted)

| Characteristic | Symbol | Min | Typ | Max | Unit |
|--|------------------------------------|---|----------------|--|---------------|
| Input High Voltage Logic, Q, RESET E | V_{IH} V_{IHR} V_{IHC} | $V_{SS} + 2.0$ $V_{SS} + 4.0$ $V_{CC} - 0.75$ | — — — | V_{CC} V_{CC} $V_{CC} + 0.3$ | V |
| Input Low Voltage Logic, RESET E Q | V_{IL} V_{ILC} V_{ILQ} | $V_{SS} - 0.3$ $V_{SS} - 0.3$ $V_{SS} - 0.3$ | — — — | $V_{SS} + 0.8$ $V_{SS} + 0.4$ $V_{SS} + 0.6$ | V |
| Input Leakage Current ($V_{in} = 0$ to 5.25 V , $V_{CC} = \text{max}$) Logic, Q, RESET E | I_{in} | — — | — — | 2.5 100 | μA |
| dc Output High Voltage ($I_{Load} = -205 \mu\text{A}$, $V_{CC} = \text{min}$) ($I_{Load} = -145 \mu\text{A}$, $V_{CC} = \text{min}$) ($I_{Load} = -100 \mu\text{A}$, $V_{CC} = \text{min}$) D0-D7 A0-A15, R/W BA, BS, LIC, AVMA, BUSY | V_{OH} | $V_{SS} + 2.4$ $V_{SS} + 2.4$ $V_{SS} + 2.4$ | — — — | — — — | V |
| dc Output Low Voltage ($I_{Load} = 2.0 \text{ mA}$, $V_{CC} = \text{min}$) | V_{OL} | — | — | $V_{SS} + 0.5$ | V |
| Internal Power Dissipation (Measured at $T_A = 0^\circ\text{C}$ in Steady State Operation) | P_{INT} | — | — | 1.0 | W |
| Capacitance ($V_{in} = 0$, $T_A = 25^\circ\text{C}$, $f = 1.0 \text{ MHz}$) D0-D7, Logic Inputs, Q, RESET E A0-A15, R/W, BA, BS, LIC, AVMA, BUSY | C_{in} C_{out} | — — — | 10 30 10 | 15 50 15 | pF |
| Frequency of Operation (E and Q Inputs) MC6809E MC68A09E MC68B09E | f | 0.1 0.1 0.1 | — — — | 1.0 1.5 2.0 | MHz |
| Hi-Z (Off State) Input Current ($V_{in} = 0.4$ to 2.4 V , $V_{CC} = \text{max}$) D0-D7 A0-A15, R/W | I_{TSI} | — — | 2.0 — | 10 100 | μA |

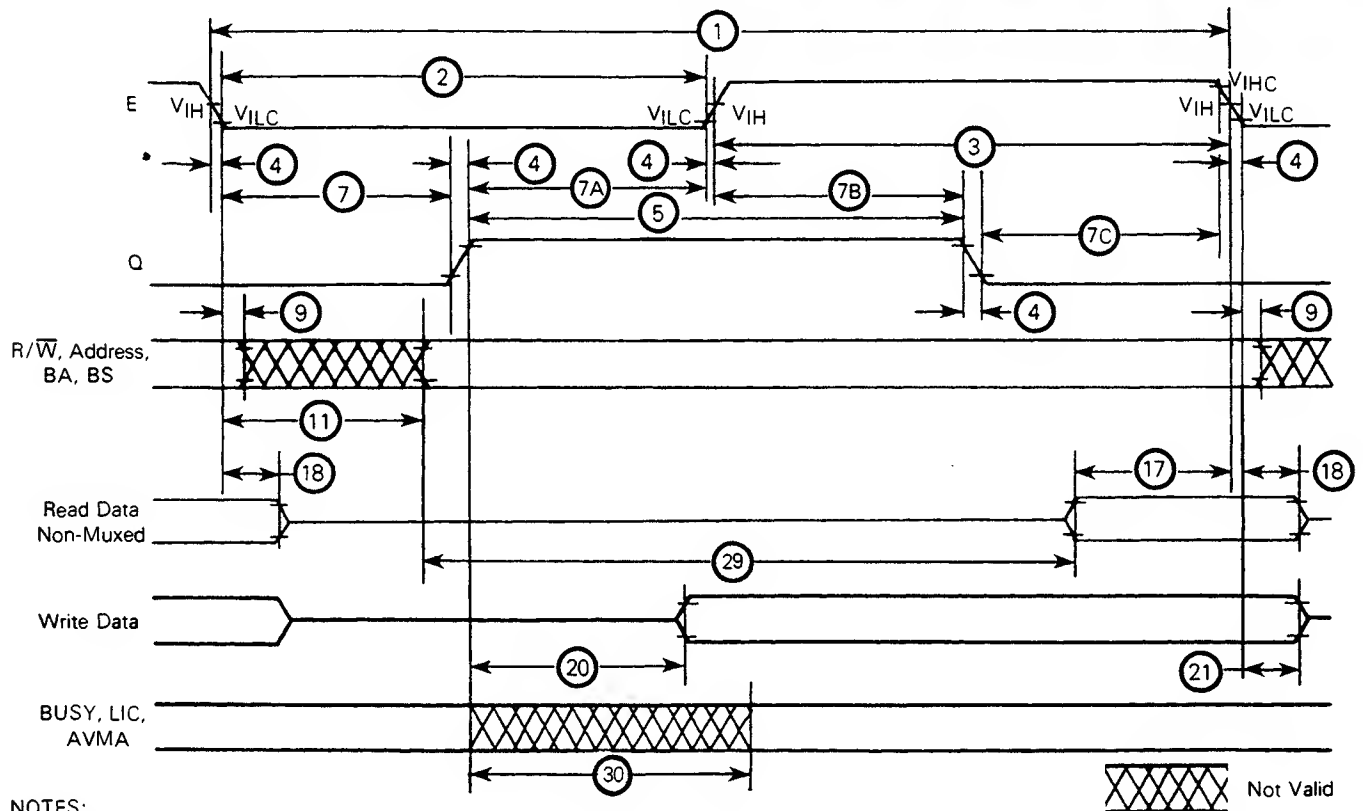
* Capacitances are periodically tested rather than 100% tested.



MOTOROLA Semiconductor Products Inc.

BUS TIMING CHARACTERISTICS (See Notes 1, 2, 3, and 4)

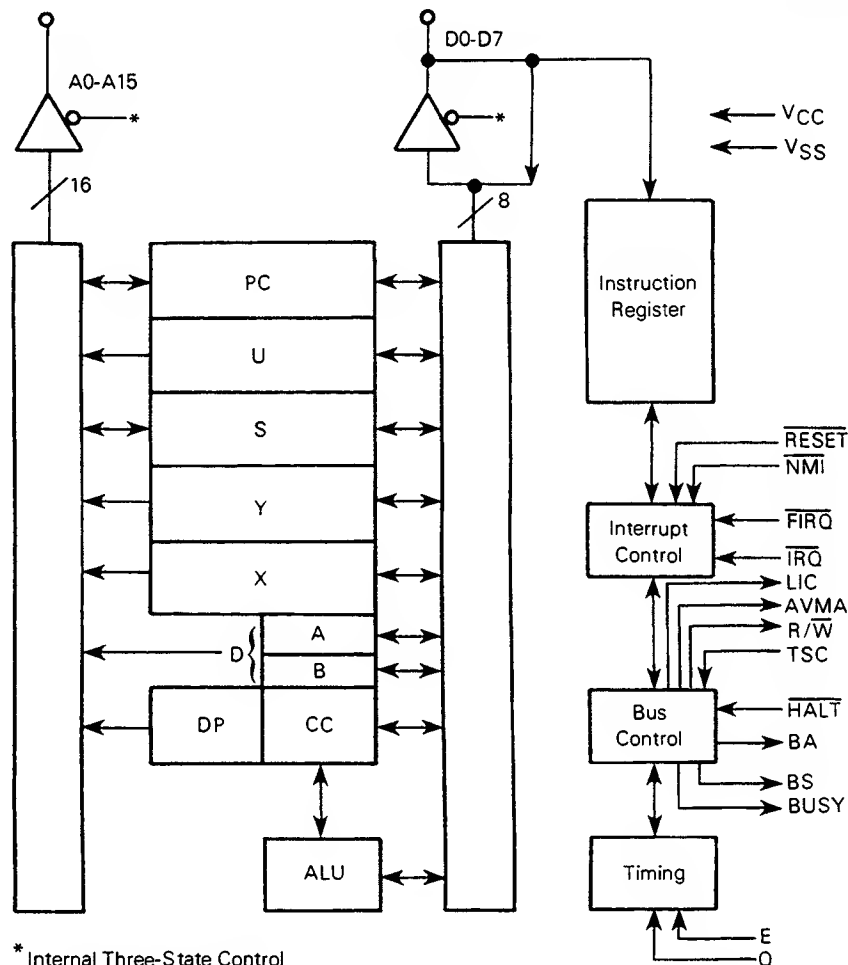
| Ident. Number | Characteristics | Symbol | MC6809E | | MC68A09E | | MC68B09E | | Unit |
|---------------|--|--------------------|---------|------|----------|------|----------|------|---------|
| | | | Min | Max | Min | Max | Min | Max | |
| 1 | Cycle Time | t_{cyc} | 1.0 | 10 | 0.667 | 10 | 0.5 | 10 | μs |
| 2 | Pulse Width, E Low | PW_{EL} | 450 | 9500 | 295 | 9500 | 210 | 9500 | ns |
| 3 | Pulse Width, E High | PW_{EH} | 450 | 9500 | 280 | 9500 | 220 | 9500 | ns |
| 4 | Clock Rise and Fall Time | t_r, t_f | — | 25 | — | 25 | — | 20 | ns |
| 5 | Pulse Width, Q High | PW_{QH} | 450 | 9500 | 280 | 9500 | 220 | 9500 | ns |
| 7 | Delay Time, E to Q Rise | t_{EQ1} | 200 | — | 130 | — | 100 | — | ns |
| 7A | Delay Time, Q High to E Rise | t_{EQ2} | 200 | — | 130 | — | 100 | — | ns |
| 7B | Delay Time, E High to Q Fall | t_{EQ3} | 200 | — | 130 | — | 100 | — | ns |
| 7C | Delay Time, Q High to E Fall | t_{EQ4} | 200 | — | 130 | — | 100 | — | ns |
| 9 | Address Hold Time | t_{AH} | 20 | — | 20 | — | 20 | — | ns |
| 11 | Address Delay Time from E Low (BA, BS, R/W) | t_{AD} | — | 200 | — | 140 | — | 110 | ns |
| 17 | Read Data Setup Time | t_{DSR} | 80 | — | 60 | — | 40 | — | ns |
| 18 | Read Data Hold Time | t_{DHR} | 10 | — | 10 | — | 10 | — | ns |
| 20 | Data Delay Time from Q | t_{DDQ} | — | 200 | — | 140 | — | 110 | ns |
| 21 | Write Data Hold Time | t_{DHW} | 30 | — | 30 | — | 30 | — | ns |
| 29 | Usable Access Time | t_{ACC} | 695 | — | 440 | — | 330 | — | ns |
| 30 | Control Delay Time | t_{CD} | — | 300 | — | 250 | — | 200 | ns |
| | Interrupts, HALT, RESET, and TSC Setup Time (Figures 6, 7, 8, 9, 12, and 13) | t_{PCS} | 200 | — | 140 | — | 110 | — | ns |
| | TSC Drive to Valid Logic Level (Figure 13) | t_{TSV} | — | 210 | — | 150 | — | 120 | ns |
| | TSC Release MOS Buffers to High Impedance (Figure 13) | t_{TSR} | — | 200 | — | 140 | — | 110 | ns |
| | TSC Hi-Z Delay Time (Figure 13) | t_{TSD} | — | 120 | — | 85 | — | 80 | ns |
| | Processor Control Rise and Fall Time (Figure 7) | t_{PCr}, t_{PCf} | — | 100 | — | 100 | — | 100 | ns |

FIGURE 1 — READ/WRITE DATA TO MEMORY OR PERIPHERALS TIMING DIAGRAM

NOTES:

1. Voltage levels shown are $V_L \leq 0.4 V$, $V_H \geq 2.4 V$, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.
3. Hold time (9) for BA and BS is not specified.
4. Usable access time is computed by: $1 - 4 - 11 \text{ max} - 17$.


MOTOROLA Semiconductor Products Inc.

FIGURE 2 — EXPANDED BLOCK DIAGRAM



PROGRAMMING MODEL

As shown in Figure 4, the MC6809E adds three registers to the set available in the MC6800. The added registers include a direct page register, the user stack pointer, and a second index register.

ACCUMULATORS (A, B, D)

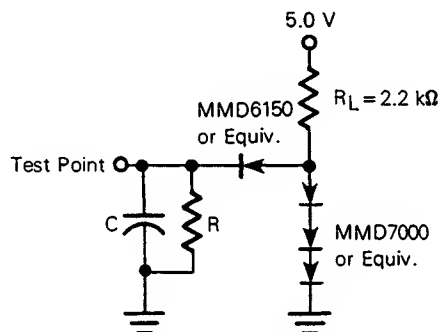
The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register, and is formed with the A register as the most significant byte.

DIRECT PAGE REGISTER (DP)

The direct page register of the MC6809E serves to enhance the direct addressing mode. The content of this register appears at the higher address outputs (A8-A15) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure M6800 compatibility, all bits of this register are cleared during processor reset.

FIGURE 3 — BUS TIMING TEST LOAD

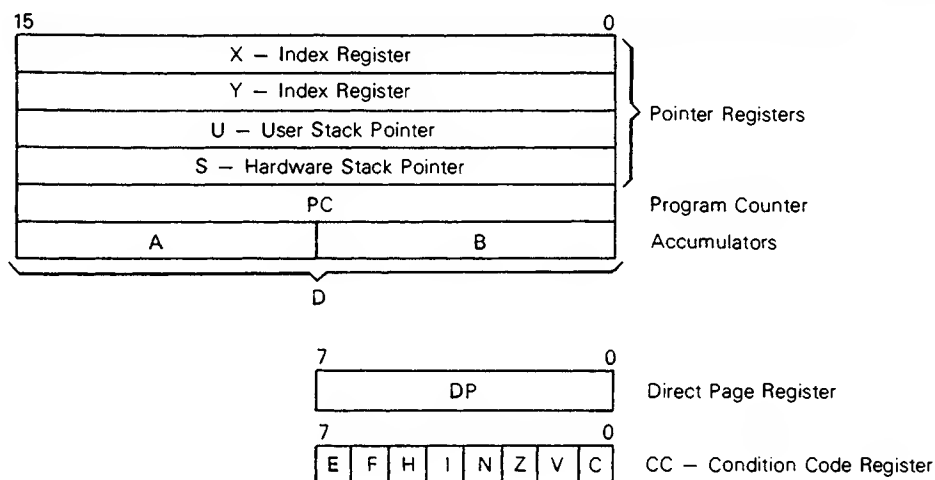


C = 30 pF for BA, BS, LIC, AVMA, BUSY
130 pF for D0-D7
90 pF for A0-A15, R/W

R = 11.7 kΩ for D0-D7
16.5 kΩ for A0-A15, R/W
24 kΩ for BA, BS, LIC, AVMA, BUSY



FIGURE 4 — PROGRAMMING MODEL OF THE MICROPROCESSING UNIT



INDEX REGISTERS (X, Y)

The index registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented and decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

STACK POINTER (U, S)

The hardware stack pointer (S) is used automatically by the processor during subroutine calls and interrupts. The user stack pointer (U) is controlled exclusively by the programmer. This allows arguments to be passed to and from subroutines with ease. The U register is frequently used as a stack marker. Both stack pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support **Push** and **Pull** instructions. This allows the MC6809E to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

NOTE

The stack pointers of the MC6809E point to the top of the stack in contrast to the MC6800 stack pointer, which pointed to the next free location on stack.

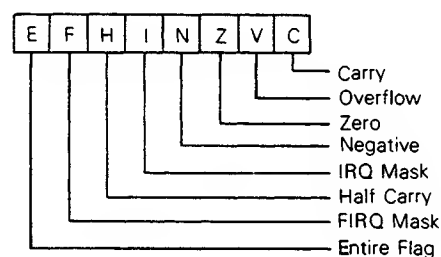
PROGRAM COUNTER

The program counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative addressing is provided allowing the program counter to be used like an index register in some situations.

CONDITION CODE REGISTER

The condition code register defines the state of the processor at any given time. See Figure 4.

FIGURE 5 — CONDITION CODE REGISTER FORMAT



CONDITION CODE REGISTER DESCRIPTION

BIT 0 (C)

Bit 0 is the carry flag and is usually the carry from the binary ALU. C is also used to represent a "borrow" from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

BIT 1 (V)

Bit 1 is the overflow flag and is set to a one by an operation which causes a signed twos complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

BIT 2 (Z)

Bit 2 is the zero flag and is set to a one if the result of the previous operation was identically zero.



BIT 3 (N)

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative twos complement result will leave N set to a one.

BIT 4 (I)

Bit 4 is the $\overline{\text{IRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{IRQ}}$ line if this bit is set to a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$, and $\overline{\text{SWI}}$ all set I to a one. $\overline{\text{SWI2}}$ and $\overline{\text{SWI3}}$ do not affect I.

BIT 5 (H)

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

BIT 6 (F)

Bit 6 is the $\overline{\text{FIRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{FIRQ}}$ line if this bit is a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{SWI}}$, and $\overline{\text{RESET}}$ all set F to a one. $\overline{\text{IRQ}}$, $\overline{\text{SWI2}}$, and $\overline{\text{SWI3}}$ do not affect F.

BIT 7 (E)

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the condition code register represents past action.

PIN DESCRIPTIONS**POWER (V_{SS} , V_{CC})**

Two pins are used to supply power to the part: V_{SS} is ground or 0 volts, while V_{CC} is $+5.0 \text{ V} \pm 5\%$.

ADDRESS BUS (A0-A15)

Sixteen pins are used to output address information from the MPU onto the address bus. When the processor does not require the bus for a data transfer, it will output address FFFF_{16} , $\text{R}/\overline{\text{W}} = 1$, and $\text{BS} = 0$; this is a "dummy access" or VMA cycle. All address bus drivers are made high-impedance when output bus available (BA) is high or when TSC is asserted. Each pin will drive one Schottky TTL load or four LSTTL loads and 90 pF.

DATA BUS (D0-D7)

These eight pins provide communication with the system bidirectional data bus. Each pin will drive one Schottky TTL load or four LSTTL loads and 130 pF.

READ/WRITE ($\text{R}/\overline{\text{W}}$)

This signal indicates the direction of data transfer on the data bus. A low indicates that the MPU is writing data onto the data bus. $\text{R}/\overline{\text{W}}$ is made high impedance when BA is high or when TSC is asserted.

 $\overline{\text{RESET}}$

A low level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 6. The

reset vectors are fetched from locations FFFE_{16} and FFFF_{16} (Table 1) when interrupt acknowledge is true, ($\text{BA} \cdot \text{BS} = 1$). During initial power on, the reset line should be held low until the clock input signals are fully operational.

Because the MC6809E $\overline{\text{RESET}}$ pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the processor.

 $\overline{\text{HALT}}$

A low level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven high indicating the buses are high impedance. BS is also high which indicates the processor is in the halt state. While halted, the MPU will not respond to external real-time requests ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) although $\overline{\text{NMI}}$ or $\overline{\text{RESET}}$ will be latched for later response. During the halt state, Q and E should continue to run normally. A halted state ($\text{BA} \cdot \text{BS} = 1$) can be achieved by pulling $\overline{\text{HALT}}$ low while $\overline{\text{RESET}}$ is still low. See Figure 7.

BUS AVAILABLE, BUS STATUS (BA, BS)

The bus available output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. When BA goes low, a dead cycle will elapse before the MPU acquires the bus. BA will not be asserted when TSC is active, thus allowing dead cycle consistency.

The bus status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

| MPU State | | MPU State Definition |
|-----------|----|--------------------------------|
| BA | BS | |
| 0 | 0 | Normal (Running) |
| 0 | 1 | Interrupt or Reset Acknowledge |
| 1 | 0 | Sync Acknowledge |
| 1 | 1 | Halt Acknowledge |

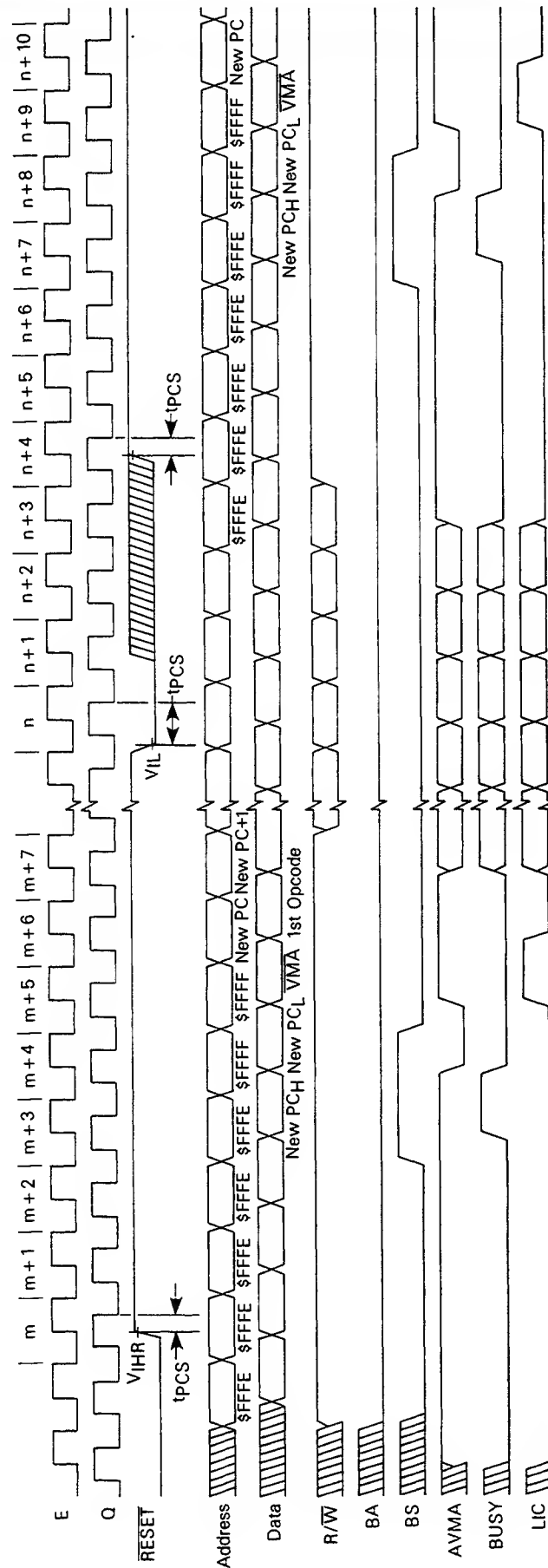
Interrupt Acknowledge is indicated during both cycles of a hardware vector fetch ($\overline{\text{RESET}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, $\overline{\text{SWI}}$, $\overline{\text{SWI2}}$, $\overline{\text{SWI3}}$). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

TABLE 1 — MEMORY MAP FOR INTERRUPT VECTORS

| Memory Map For Vector Locations | | Interrupt Vector Description |
|---------------------------------|------|------------------------------|
| MS | LS | |
| FFFE | FFFF | $\overline{\text{RESET}}$ |
| FFFC | FFFD | $\overline{\text{NMI}}$ |
| FFFA | FFFB | $\overline{\text{SWI}}$ |
| FFF8 | FFF9 | $\overline{\text{IRQ}}$ |
| FFF6 | FFF7 | $\overline{\text{FIRQ}}$ |
| FFF4 | FFF5 | $\overline{\text{SWI2}}$ |
| FFF2 | FFF3 | $\overline{\text{SWI3}}$ |
| FFF0 | FFF1 | Reserved |

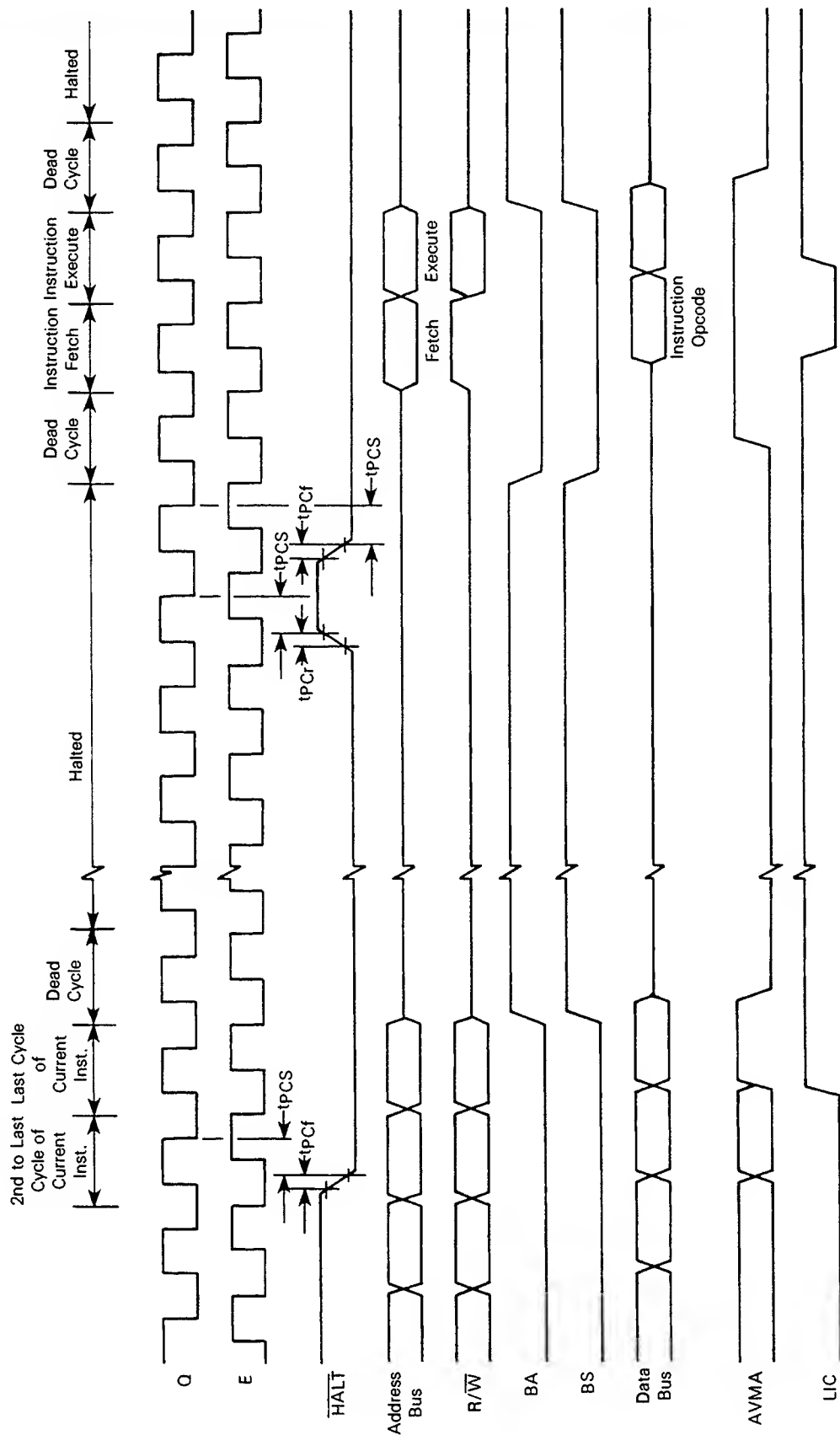


FIGURE 6 – RESET TIMING



NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.



FIGURE 7 — HALT AND SINGLE INSTRUCTION EXECUTION TIMING FOR SYSTEM DEBUG

NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.



Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt Acknowledge is indicated when the MC6809E is in a halt condition.

NON MASKABLE INTERRUPT ($\overline{\text{NMI}}$)*

A negative transition on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program and also has a higher priority than $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, or software interrupts. During recognition of an $\overline{\text{NMI}}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{\text{NMI}}$ will not be recognized until the first program load of the hardware stack pointer (S). The pulse width of $\overline{\text{NMI}}$ low must be at least one E cycle. If the $\overline{\text{NMI}}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 8.

FAST-INTERRUPT REQUEST ($\overline{\text{FIRQ}}$)*

A low level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard interrupt request ($\overline{\text{IRQ}}$) and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 9.

INTERRUPT REQUEST ($\overline{\text{IRQ}}$)*

A low level input on this pin will initiate an interrupt request sequence provided the mask bit (I) in the CC is clear. Since $\overline{\text{IRQ}}$ stacks the entire machine state, it provides a slower response to interrupts than $\overline{\text{FIRQ}}$. $\overline{\text{IRQ}}$ also has a lower priority than $\overline{\text{FIRQ}}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 8.

CLOCK INPUTS E, Q

E and Q are the clock signals required by the MC6809E. Q must lead E; that is, a transition on Q must be followed by a similar transition on E after a minimum delay. Addresses will be valid from the MPU, t_{AD} after the falling edge of E, and data will be latched from the bus by the falling edge of E. While the Q input is fully TTL compatible, the E input directly drives internal MOS circuitry and, thus, requires a high level above normal TTL levels. This approach minimizes clock skew inherent with an internal buffer. Refer to **BUS TIMING CHARACTERISTICS** for E and Q and to Figure 10 which shows a simple clock generator for the MC6809E.

BUSY

BUSY will be high for the read and modify cycles of a read-modify-write instruction and during the access of the first byte of a double-byte operation (e.g., LDX, STD, ADDD). BUSY is also high during the first byte of any indirect or other vector fetch (e.g., jump extended, SWI indirect, etc.).

In a multiprocessor system, BUSY indicates the need to

defer the re arbitration of the next bus cycle to insure the integrity of the above operations. This difference provides the indivisible memory access required for a "test-and-set" primitive, using any one of several read-modify-write instructions.

BUSY does not become active during PSH or PUL operations. A typical read-modify-write instruction (ASL) is shown in Figure 11. Timing information is given in Figure 12. BUSY is valid t_{CD} after the rising edge of Q.

AVMA

AVMA is the advanced VMA signal and indicates that the MPU will use the bus in the following bus cycle. The predictive nature of the AVMA signal allows efficient shared-bus multiprocessor systems. AVMA is low when the MPU is in either a HALT or SYNC state. AVMA is valid t_{CD} after the rising edge of Q.

LIC

LIC (last instruction cycle) is high during the last cycle of every instruction, and its transition from high to low will indicate that the first byte of an opcode will be latched at the end of the present bus cycle. LIC will be high when the MPU is halted at the end of an instruction (i.e., not in CWAI or RESET), in sync state, or while stacking during interrupts. LIC is valid t_{CD} after the rising edge of Q.

TSC

TSC (three-state control) will cause MOS address, data, and R/W buffers to assume a high-impedance state. The control signals (BA, BS, BUSY, AVMA, and LIC) will not go to the high-impedance state. TSC is intended to allow a single bus to be shared with other bus masters (processors or DMA controllers).

While E is low, TSC controls the address buffers and R/W directly. The data bus buffers during a write operation are in a high-impedance state until Q rises at which time, if TSC is true, they will remain in a high-impedance state. If TSC is held beyond the rising edge of E, then it will be internally latched, keeping the bus drivers in a high-impedance state for the remainder of the bus cycle. See Figure 13.

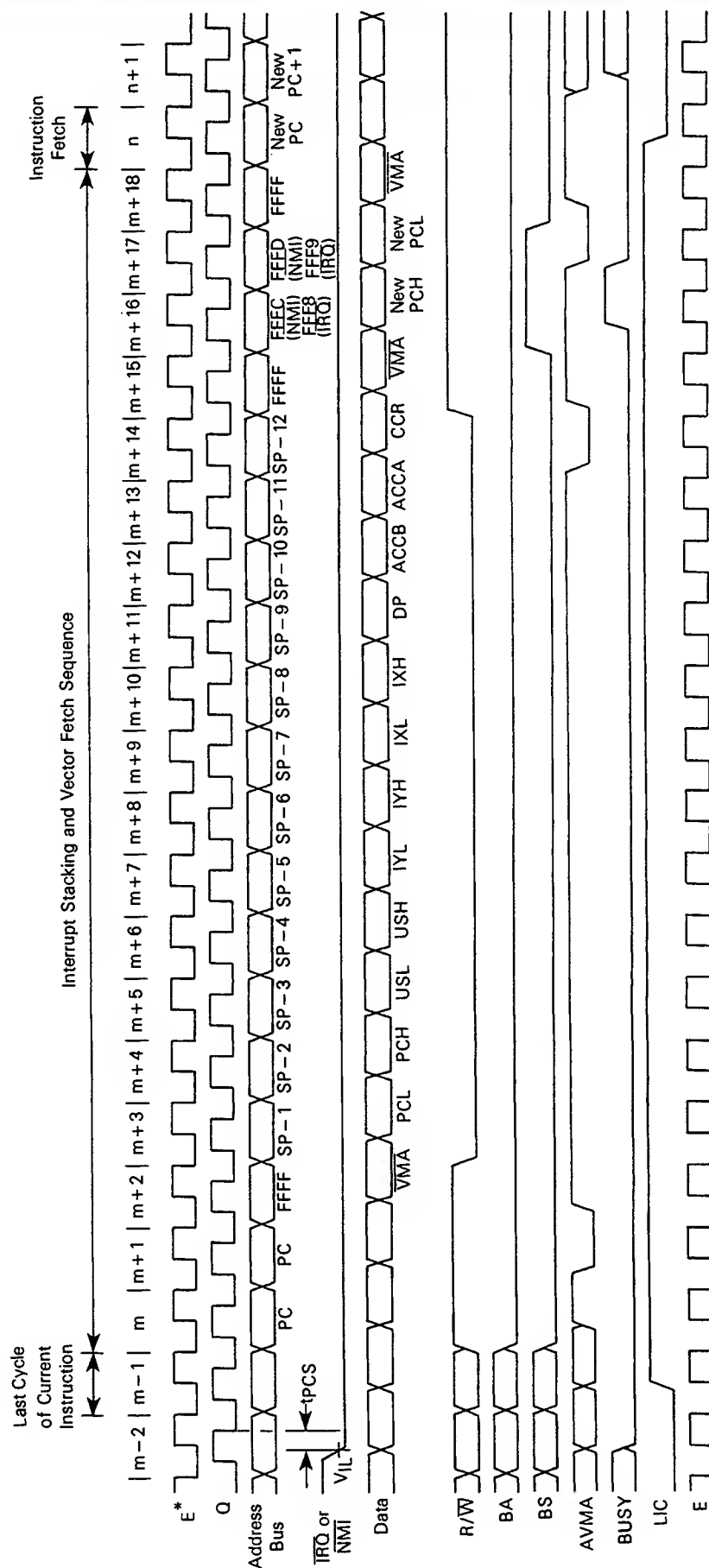
MPU OPERATION

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins after RESET and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI, and SYNC. An interrupt or HALT input can also alter the normal execution of instructions. Figure 14 is the flowchart for the MC6809E.

* $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, and $\overline{\text{IRQ}}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{\text{IRQ}}$ and $\overline{\text{FIRQ}}$ do not remain low until completion of the current instruction, they may not be recognized. However, $\overline{\text{NMI}}$ is latched and need only remain low for one cycle. No interrupts are recognized or latched between the falling edge of RESET and the rising edge of BS indicating RESET acknowledge. See RESET sequence in the MPU flowchart in Figure 14.

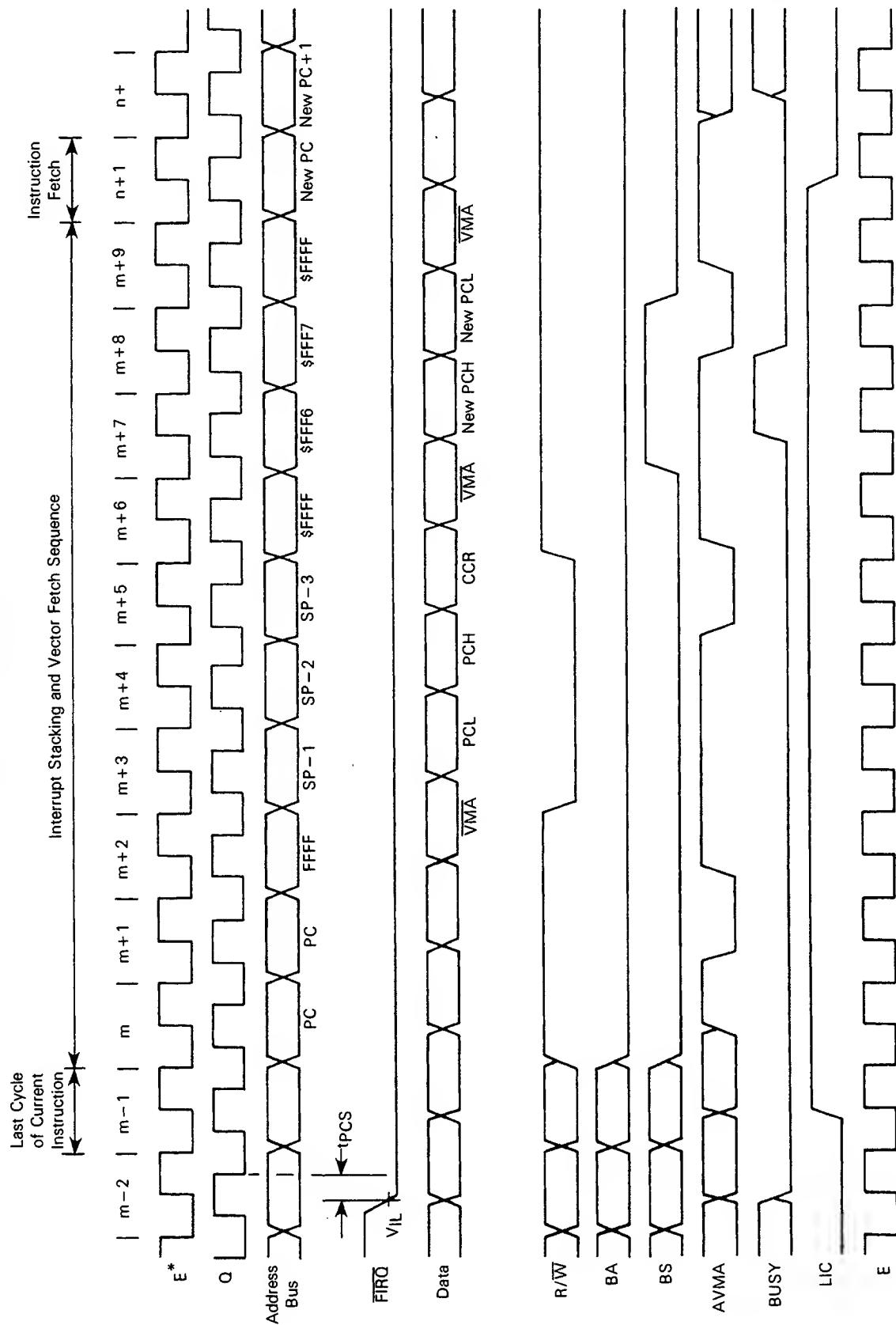


FIGURE 8 – $\overline{\text{IRQ}}$ AND $\overline{\text{NMI}}$ INTERRUPT TIMING



* E clock shown for reference only.

NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

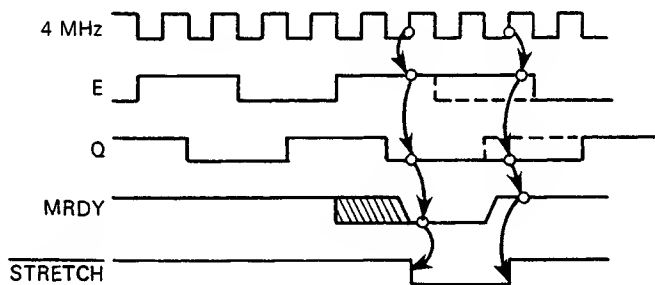
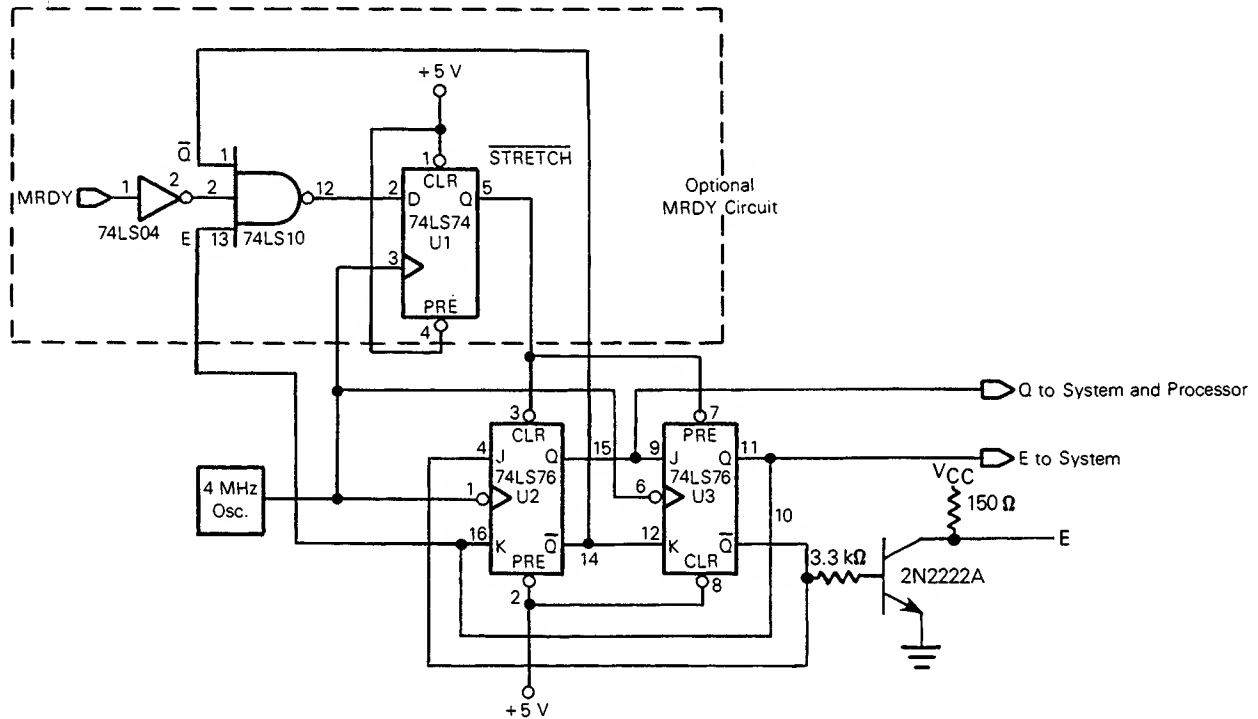
FIGURE 9 — $\overline{\text{FIRQ}}$ INTERRUPT TIMING

* E clock shown for reference only.

NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.



FIGURE 10 — CLOCK GENERATOR



NOTE: If optional circuit is not included the CLR and PRE inputs of U2 and U3 must be tied high.

FIGURE 11 — READ-MODIFY-WRITE INSTRUCTION EXAMPLE (ASL EXTENDED INDIRECT)

| Memory Location | Memory Contents | Contents Description |
|-----------------|-----------------|----------------------------|
| PC → \$0200 | \$68 | ASL Indexed Opcode |
| \$0201 | \$9F | Extended Indirect Postbyte |
| \$0202 | \$63 | Indirect Address Hi-Byte |
| \$0203 | \$00 | Indirect Address Lo-Byte |
| \$0204 | | Next Main Instruction |
| \$6300 | \$E3 | Effective Address Hi-Byte |
| \$6301 | \$D6 | Effective Address Lo-Byte |
| \$E3D6 | \$5C | Target Data |



FIGURE 12 — BUSY TIMING

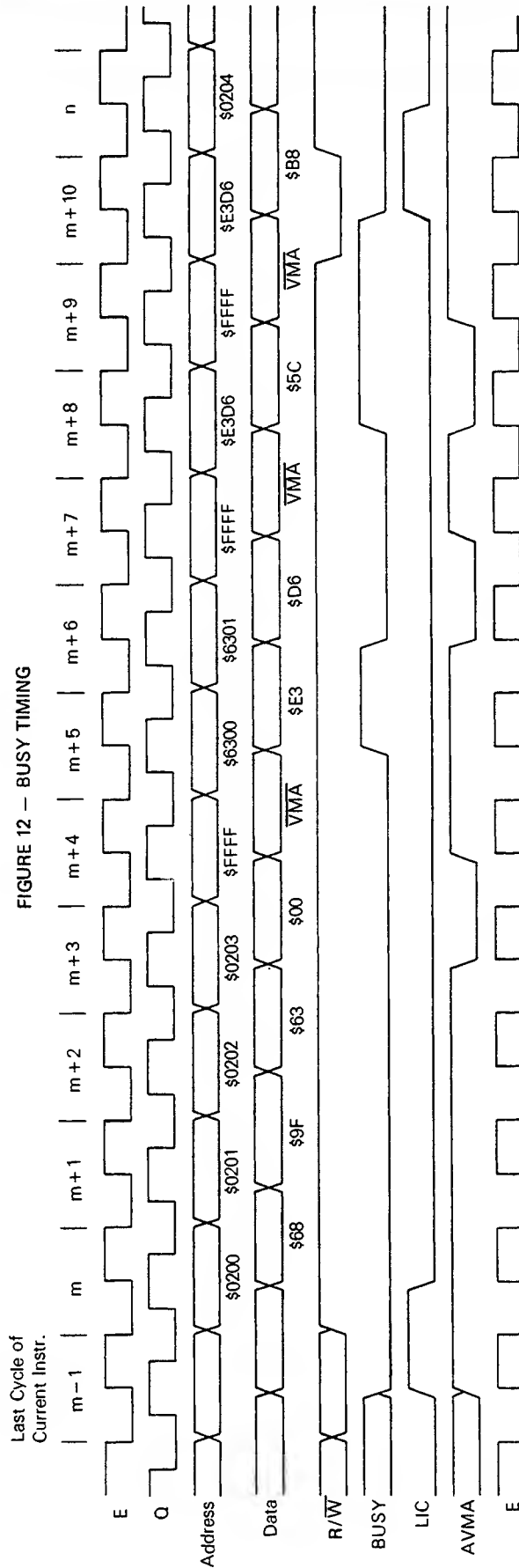
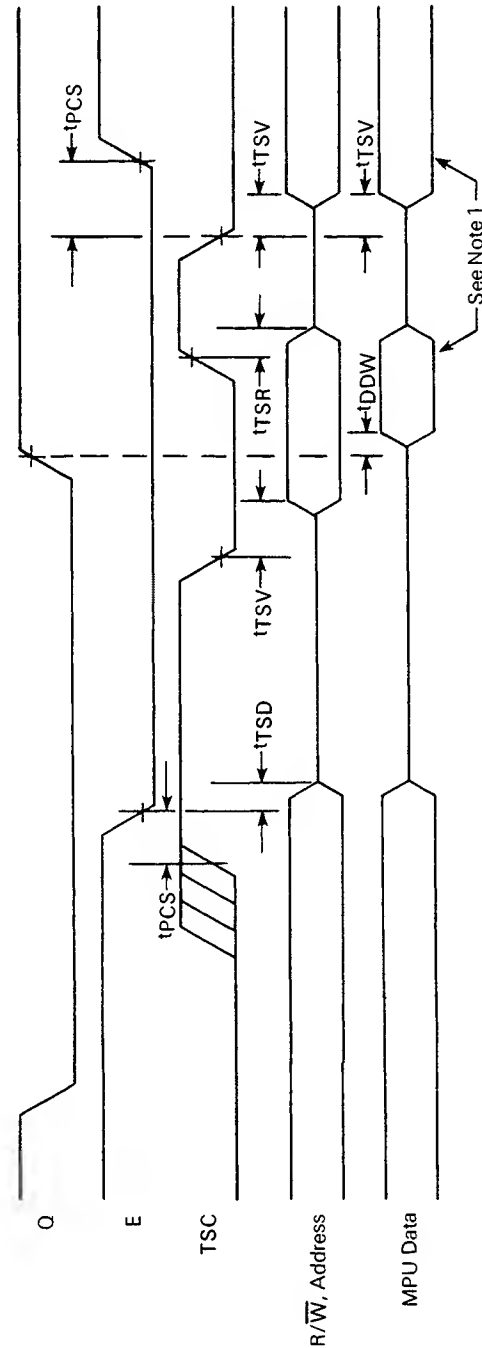


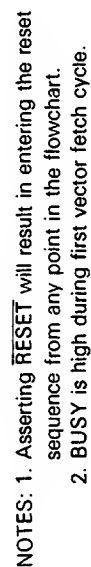
FIGURE 13 — TSC TIMING



NOTES:

1. Data will be asserted by the MPU only during the interval while R/\overline{W} is low and (E or Q) is high. A composite bus cycle is shown to give most cases of timing.
2. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.





| Bus State | BA | BS |
|--------------------------------|----|----|
| Running | 0 | 0 |
| Interrupt or Reset Acknowledge | 0 | 1 |
| Sync Acknowledge | 1 | 0 |
| Halt Acknowledge | 1 | 1 |

ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The MC6809E has the most complete set of addressing modes available on any microcomputer today. For example, the MC6809E has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the MC6809E:

- Inherent (Includes Accumulator)
- Immediate
- Extended
 - Extended Indirect
- Direct
- Register
- Indexed
 - Zero-Offset
 - Constant Offset
 - Accumulator Offset
 - Auto Increment/Decrement
 - Indexed Indirect
- Relative
 - Short/Long Relative Branching
 - Program Counter Relative Addressing

INHERENT (INCLUDES ACCUMULATOR)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of inherent addressing are: ABX, DAA, SWI, ASRA, and CLRB.

IMMEDIATE ADDRESSING

In immediate addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately following the opcode of the instruction). The MC6809E uses both 8- and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with immediate addressing are:

```
LDA  #$20
LDX  #$F000
LDY  #CAT
```

NOTE

signifies immediate addressing; \$ signifies hexadecimal value to the MC6809 assembler.

EXTENDED ADDRESSING

In extended addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of extended addressing include:

```
LDA  CAT
STX  MOUSE
LDD  $2000
```

EXTENDED INDIRECT

As a special case of indexed addressing (discussed below), one level of indirection may be added to extended addressing. In extended indirect, the two bytes following the postbyte of an indexed instruction contain the address of the data.

```
LDA  [CAT]
LDX  [$FFFE]
STU  [DOG]
```

DIRECT ADDRESSING

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower eight bits of the address to be used. The upper eight bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to \$00 on reset, direct addressing on the MC6809E is upward compatible with direct addressing on the M6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA  where DP = $00
LDB  where DP = $10
LDD  < CAT
```

NOTE

< is an assembler directive which forces direct addressing.

REGISTER ADDRESSING

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

| | | |
|------|------------|---------------------------------|
| TFR | X, Y | Transfers X into Y |
| EXG | A, B | Exchanges A with B |
| PSHS | A, B, X, Y | Push Y, X, B and A onto S stack |
| PULU | X, Y, D | Pull D, X, and Y from U stack |

INDEXED ADDRESSING

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode, as well as the pointer register to be used. Figure 15 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.



FIGURE 15 — INDEXED ADDRESSING POSTBYTE REGISTER BIT ASSIGNMENTS

| Post-Byte Register Bit | | | | | | | | Indexed Addressing Mode |
|------------------------|---|---|---|---|---|---|---|--------------------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 0 | R | R | d | d | d | d | d | EA = ,R + 5 Bit Offset |
| 1 | R | R | 0 | 0 | 0 | 0 | 0 | ,R + |
| 1 | R | R | i | 0 | 0 | 0 | 1 | ,R + + |
| 1 | R | R | 0 | 0 | 0 | 1 | 0 | , - R |
| 1 | R | R | i | 0 | 0 | 1 | 1 | , - - R |
| 1 | R | R | i | 0 | 1 | 0 | 0 | EA = ,R + 0 Offset |
| 1 | R | R | i | 0 | 1 | 0 | 1 | EA = ,R + ACCB Offset |
| 1 | R | R | i | 0 | 1 | 1 | 0 | EA = ,R + ACCA Offset |
| 1 | R | R | i | 1 | 0 | 0 | 0 | EA = ,R + 8 Bit Offset |
| 1 | R | R | i | 1 | 0 | 0 | 1 | EA = ,R + 16 Bit Offset |
| 1 | R | R | i | 1 | 0 | 1 | 1 | EA = ,R + D Offset |
| 1 | x | x | i | 1 | 1 | 0 | 0 | EA = ,PC + 8 Bit Offset |
| 1 | x | x | i | 1 | 1 | 0 | 1 | EA = ,PC + 16 Bit Offset |
| 1 | R | R | i | 1 | 1 | 1 | 1 | EA = [,Address] |

Addressing Mode Field

Indirect Field
(Sign Bit when b7 = 0)

Register Field: RR

00 = X

01 = Y

10 = U

11 = S

x = Don't Care
d = Offset Bit
0 = Not Indirect
i = Indirect

ZERO-OFFSET INDEXED — In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

LDD 0, X

LDA ,S

CONSTANT OFFSET INDEXED — In this mode, twos complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offset are available:

5-bit (–16 to +15)

8-bit (–128 to +127)

16-bit (–32768 to +32767)

The twos complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The twos complement 8-bit offset is contained in a single byte following the postbyte. The twos complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

LDA 23,X

LDX –2,S

LDY 300,X

LDU CAT,Y

TABLE 2 — INDEXED ADDRESSING MODE

| Type | Forms | Non Indirect | | | | Indirect | | | |
|--|-------------------|----------------|-----------------|---|---|-------------------|-----------------|---|---|
| | | Assembler Form | Postbyte Opcode | + | + | Assembler Form | Postbyte Opcode | + | + |
| Constant Offset From R (2s Complement Offsets) | No Offset | ,R | 1RR00100 | 0 | 0 | [,R] | 1RR10100 | 3 | 0 |
| | 5-Bit Offset | n, R | 0RRnnnnn | 1 | 0 | defaults to 8-bit | | | |
| | 8-Bit Offset | n, R | 1RR01000 | 1 | 1 | [n, R] | 1RR11000 | 4 | 1 |
| | 16-Bit Offset | n, R | 1RR01001 | 4 | 2 | [n, R] | 1RR11001 | 7 | 2 |
| Accumulator Offset From R (2s Complement Offsets) | A Register Offset | A, R | 1RR00110 | 1 | 0 | [A, R] | 1RR10110 | 4 | 0 |
| | B Register Offset | B, R | 1RR00101 | 1 | 0 | [B, R] | 1RR10101 | 4 | 0 |
| | D Register Offset | D, R | 1RR01011 | 4 | 0 | [D, R] | 1RR11011 | 7 | 0 |
| Auto Increment/Decrement R | Increment By 1 | ,R + | 1RR00000 | 2 | 0 | not allowed | | | |
| | Increment By 2 | ,R + + | 1RR00001 | 3 | 0 | [,R + +] | 1RR10001 | 6 | 0 |
| | Decrement By 1 | , - R | 1RR00010 | 2 | 0 | not allowed | | | |
| | Decrement By 2 | , - - R | 1RR00011 | 3 | 0 | [, - - R] | 1RR10011 | 6 | 0 |
| Constant Offset From PC (2s Complement Offsets) | 8-Bit Offset | n, PCR | 1xx01100 | 1 | 1 | [n, PCR] | 1xx11100 | 4 | 1 |
| | 16-Bit Offset | n, PCR | 1xx01101 | 5 | 2 | [n, PCR] | 1xx11101 | 8 | 2 |
| Extended Indirect | 16-Bit Address | — | — | — | — | [n] | 10011111 | 5 | 2 |

R = X, Y, U or S
x = Don't Care

RR:

00 = X

01 = Y

10 = U

11 = S

+ and # indicate the number of additional cycles and bytes respectively for the particular indexing variation.



ACCUMULATOR-OFFSET INDEXED — This mode is similar to constant offset indexed except that the two's complement value in one of the accumulators (A, B, or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA  B, Y
LDX  D, Y
LEAX B, X
```

AUTO INCREMENT/DECREMENT INDEXED — In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used, it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or creating software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment, but the tables, etc., are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8- or 16-bit data to be accessed and is selectable by the programmer. The pre-decrement, post-increment nature of these modes allows them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA  ,X+
STD  ,Y++
LDB  ,--Y
LDX  ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

```
STX 0,X++ (X initialized to 0)
```

The desired result is to store a zero in locations \$0000 and \$0001, then increment X to point to \$0002. In reality, the following occurs:

```
0 → temp      calculate the EA; temp is a holding register
X + 2 → X      perform auto increment
X → (temp)     do store operation
```

INDEXED INDIRECT

All of the indexing modes, with the exception of auto increment/decrement by one or a ± 5 -bit offset, may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the index register and an offset.

Before Execution

```
A = XX (don't care)
X = $F000
```

```
$0100  LDA [$10,X]    EA is now $F010

$F010  $F1            $F150 is now the
$F011  $50            new EA

$F150  $AA
```

After Execution

```
A = $AA (actual data loaded)
X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA  [,X]
LDD  [10,S]
LDA  [B,Y]
LDD  [,X++]
```

RELATIVE ADDRESSING

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true, then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (one byte offset) and long (two bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address interpreted modulo 2^{16} . Some examples of relative addressing are:

```

          BEQ      CAT      (short)
          BGT      DOG      (short)
CAT       LBEQ     RAT      (long)
DOG       LBGT     RABBIT   (long)
          •
          •
          •
RAT       NOP
RABBIT    NOP
```

PROGRAM COUNTER RELATIVE

The PC can be used as the pointer register with 8- or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program counter relative addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the program counter. Examples are:

```
LDA  CAT, PCR
LEAX TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA  [CAT, PCR]
LDU  [DOG, PCR]
```



INSTRUCTION SET

The instruction set of the MC6809E is similar to that of the MC6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the new instructions are described in detail below.

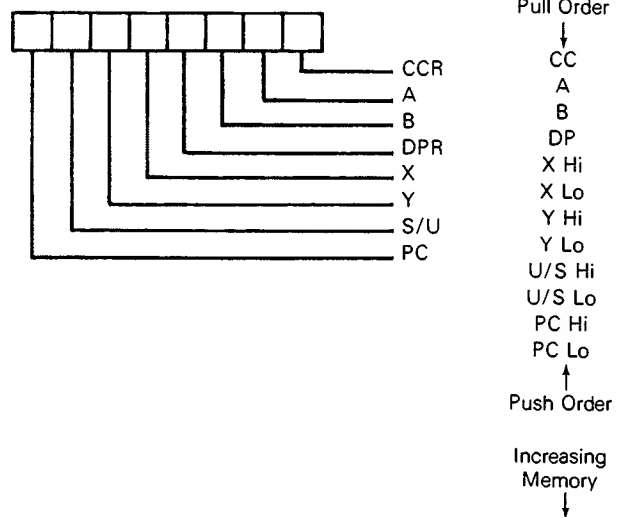
PSHU/PSHS

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register or set of registers with a single instruction.

PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual push/pull sequence is fixed; each bit defines a unique register to push or pull, as shown below.

Push/Pull Postbyte



TFR/EXG

Within the MC6809E, any register may be transferred to or exchanged with another of like size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4-7 of postbyte define the source register, while bits 0-3 represent the destination register. These are denoted as follows:

Transfer/Exchange Postbyte

| Source | Destination |
|--------|-------------|
|--------|-------------|

Register Field

| | |
|----------------|------------|
| 0000 = D (A:B) | 1000 = A |
| 0001 = X | 1001 = B |
| 0010 = Y | 1010 = CCR |
| 0011 = U | 1011 = DPR |
| 0100 = S | |
| 0101 = PC | |

NOTE

All other combinations are undefined and INVALID.

LEAX/LEAY/LEAU/LEAS

The LEA (load effective address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 3.

The LEA instruction also allows the user to access data and tables in a position independent manner. For example:

```
LEAX  MSG1, PCR
LBSR  PDATA (Print message routine)
```

```
MSG1  FCC  'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the auto increment and auto decrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

LEAa , b+ (any of the 16-bit pointer registers X, Y, U, or S may be substituted for a and b.)

1. b → temp (calculate the EA)
2. b + 1 → b (modify b, postincrement)
3. temp → a (load a)

LEAa , - b

1. b - 1 → temp (calculate EA with predecrement)
2. b - 1 → b (modify b, predecrement)
3. temp → a (load a)

TABLE 3 — LEA EXAMPLES

| Instruction | Operation | Comment |
|-------------|-------------|--------------------------------|
| LEAX 10, X | X + 10 → X | Adds 5-Bit Constant 10 to X |
| LEAX 500, X | X + 500 → X | Adds 16-Bit Constant 500 to X |
| LEAY A, Y | Y + A → Y | Adds 8-Bit A Accumulator to Y |
| LEAY D, Y | Y + D → Y | Adds 16-Bit D Accumulator to Y |
| LEAU -10, U | U - 10 → U | Subtracts 10 from U |
| LEAS -10, S | S - 10 → S | Used to Reserve Area on Stack |
| LEAS 10, S | S + 10 → S | Used to 'Clean Up' Stack |
| LEAX 5, S | S + 5 → X | Transfers As Well As Adds |



Auto increment-by-two and auto decrement-by-two instructions work similarly. Note that LEAX ,X+ does not change X; however LEAX ,−X does decrement X. LEAX 1,X should be used to increment X by one.

MUL

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

LONG AND SHORT RELATIVE BRANCHES

The MC6809E has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8- or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64K memory map. Position independent code can be easily generated through the use of relative branching. Both short (8 bit) and long (16 bit) branches are available.

SYNC

After encountering a sync instruction, the MPU enters a sync state, stops processing instructions, and waits for an interrupt. If the pending interrupt is non-maskable (NMI) or maskable (FIRQ, IRQ) with its mask bit (F or I) clear, the processor will clear the sync state and perform the normal interrupt stacking and service routine. Since FIRQ and IRQ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable (FIRQ, IRQ) with its mask bit (F or I) set, the processor will clear the sync state and continue processing by executing the next in-line instruction. Figure 16 depicts sync timing.

SOFTWARE INTERRUPTS

A software interrupt is an instruction which will cause an interrupt and its associated vector fetch. These software interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this MC6809E and are prioritized in the following order: SWI, SWI2, SWI3.

16-BIT OPERATION

The MC6809E has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes, and pulls.

CYCLE-BY-CYCLE OPERATION

The address bus cycle-by-cycle performance chart (Figure 16) illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the MC6809E. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds throughput.) Next, the operation of each opcode will follow the flowchart. VMA is an indication of FFFF₁₆ on the address bus, R/W = 1 and BS = 0. The following examples illustrate the use of the chart.

Example 1: LBSR (Branch Taken)

Before Execution SP = F000

| | | | |
|--------|-----|---|----------|
| | | • | |
| | | • | |
| | | • | |
| \$8000 | | • | LBSR CAT |
| | | • | |
| | | • | |
| \$A000 | CAT | • | |

CYCLE-BY-CYCLE FLOW

| Cycle # | Address | Data | R/W | Description |
|---------|---------|------|-----|---|
| 1 | 8000 | 17 | 1 | Opcode Fetch |
| 2 | 8001 | 20 | 1 | Offset High Byte |
| 3 | 8002 | 00 | 1 | Offset Low Byte |
| 4 | FFFF | * | 1 | VMA Cycle |
| 5 | FFFF | * | 1 | VMA Cycle |
| 6 | A000 | * | 1 | Computed Branch Address |
| 7 | FFFF | * | 1 | VMA Cycle |
| 8 | FFFF | 80 | 0 | Stack High Order Byte of Return Address |
| 9 | FFFF | 03 | 0 | Stack Low Order Byte of Return Address |

Example 2: DEC (Extended)

| | | |
|--------|-----|--------|
| \$8000 | DEC | \$A000 |
| \$A000 | FCB | \$80 |

CYCLE-BY-CYCLE FLOW

| Cycle # | Address | Data | R/W | Description |
|---------|---------|------|-----|----------------------------|
| 1 | 8000 | 7A | 1 | Opcode Fetch |
| 2 | 8001 | A0 | 1 | Operand Address, High Byte |
| 3 | 8002 | 00 | 1 | Operand Address, Low Byte |
| 4 | FFFF | * | 1 | VMA Cycle |
| 5 | A000 | 80 | 1 | Read the Data |
| 6 | FFFF | * | 1 | VMA Cycle |
| 7 | FFFF | 7F | 0 | Store the Decrement Data |

* The data bus has the data at that particular address.

INSTRUCTION SET TABLES

The instructions of the MC6809E have been broken down into five different categories. They are as follows:

8-bit operation (Table 4)

16-bit operation (Table 5)

Index register/stack pointer instructions (Table 6)

Relative branches (long or short) (Table 7)

Miscellaneous instructions (Table 8)

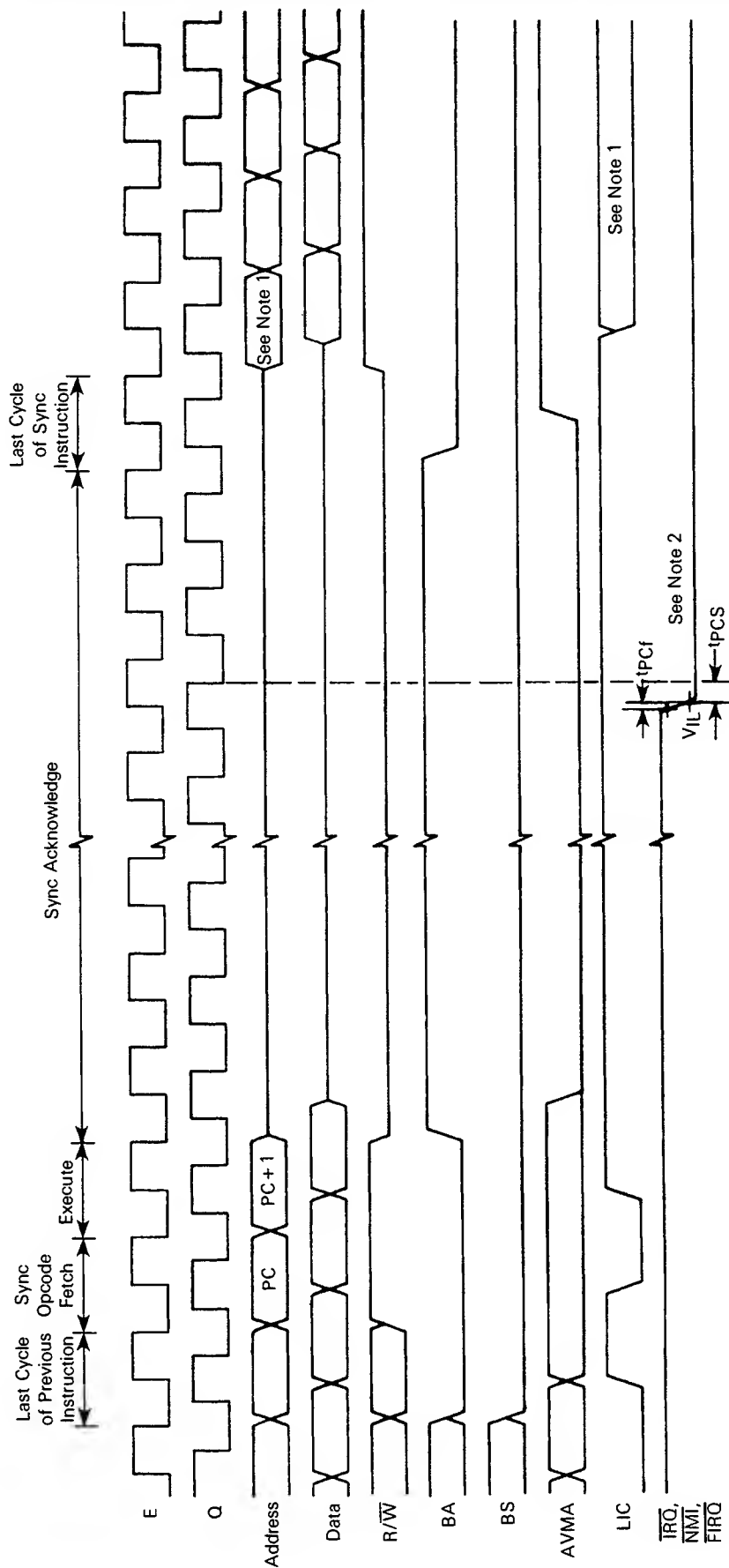
Hexadecimal values for the instructions are given in Table 9.

PROGRAMMING AID

Figure 18 contains a compilation of data that will assist you in programming the MC6809E.



FIGURE 16 — SYNC TIMING



- NOTES:
1. If the associated mask bit is set when the interrupt is requested, LIC will go low and this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted (NMI) or an unmasked FIRQ or LIC will remain high and interrupt processing will start with this cycle as in Figures 8 and 9 (Interrupt Timing).
 2. If mask bits are clear, $\overline{\text{IRQ}}$ and $\overline{\text{FIRQ}}$ must be held low for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.
 3. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.



FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 1 of 5)

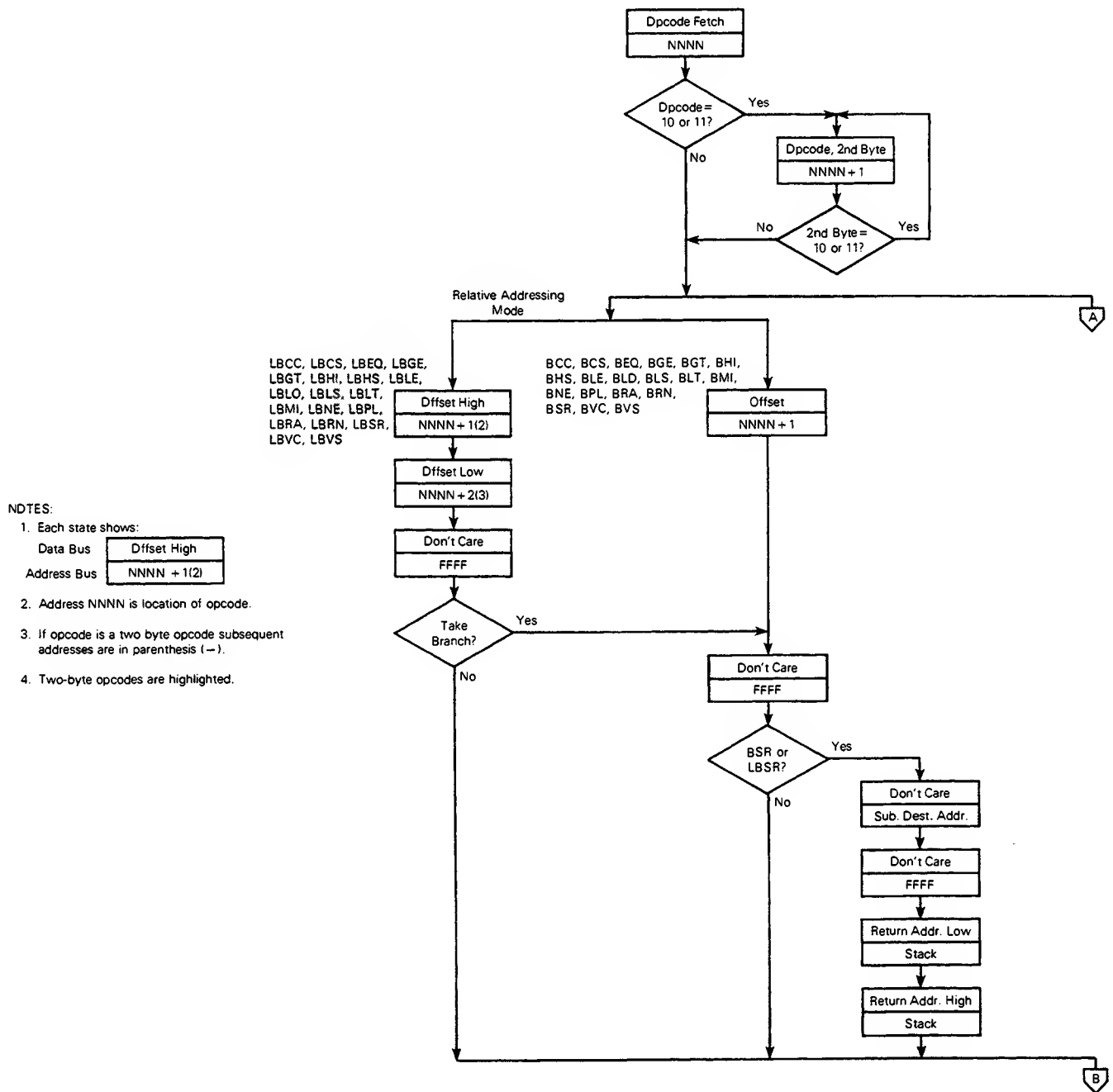


FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 2 of 5)

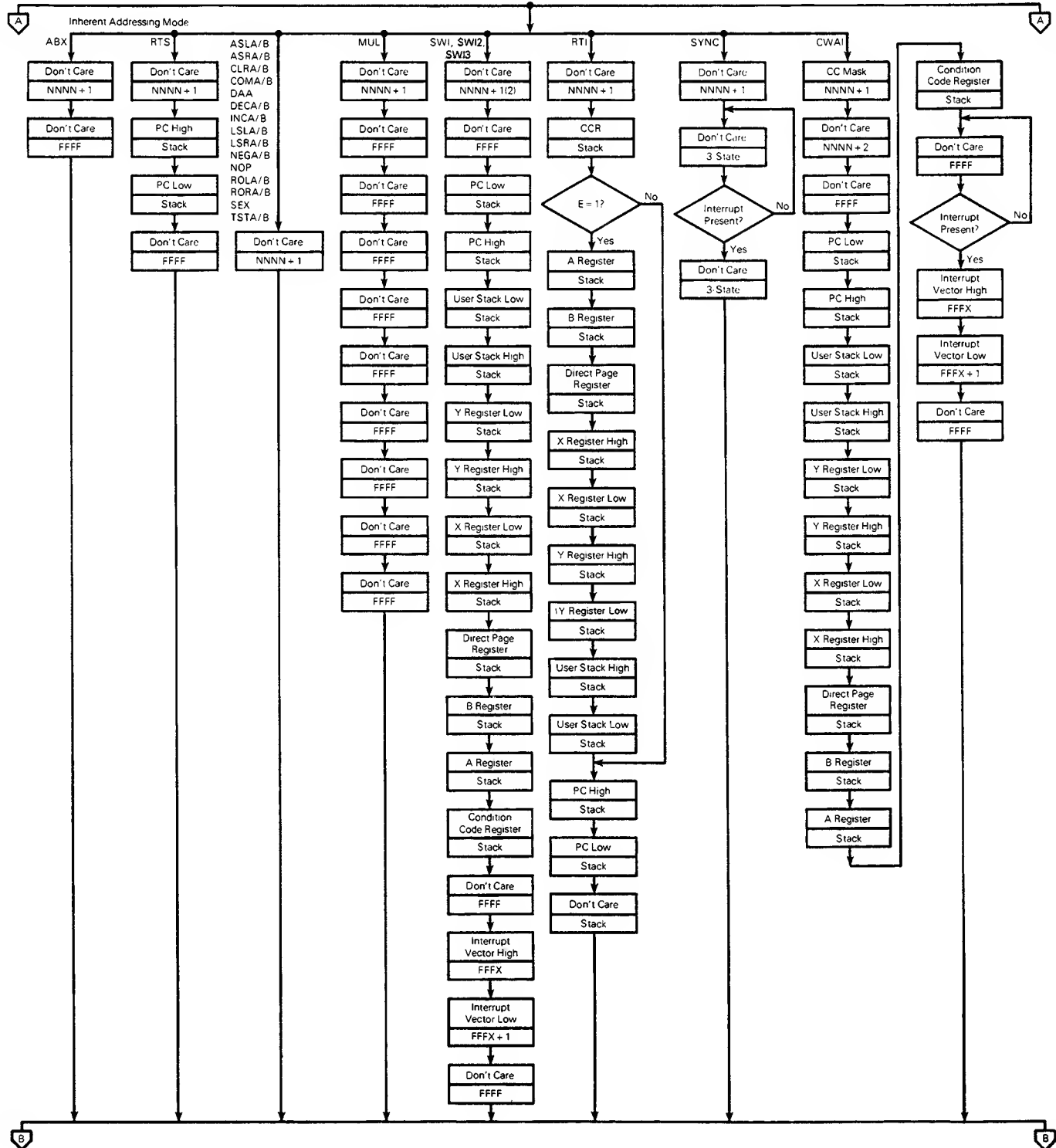


FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 3 of 5)

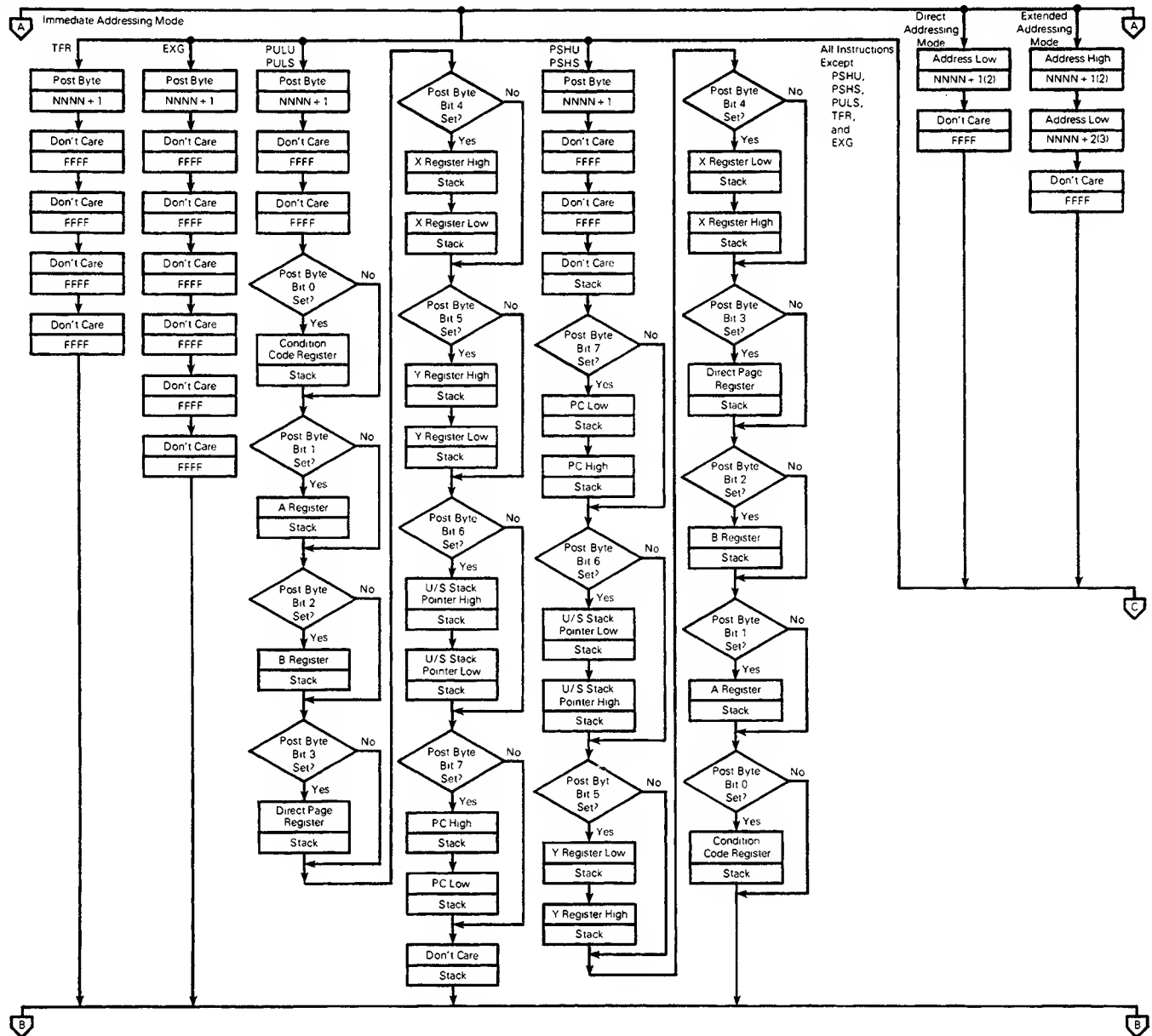


FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 4 of 5)

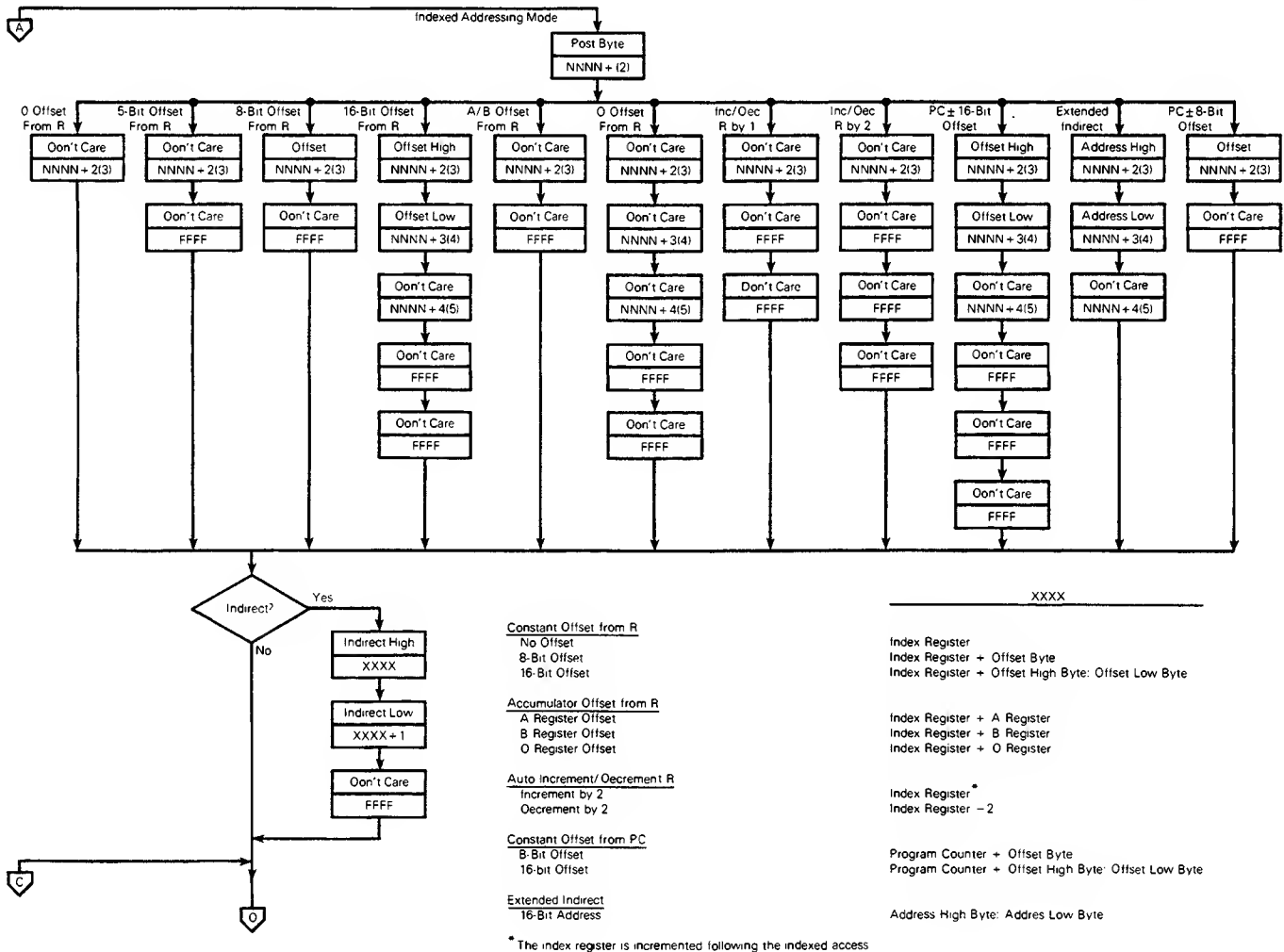
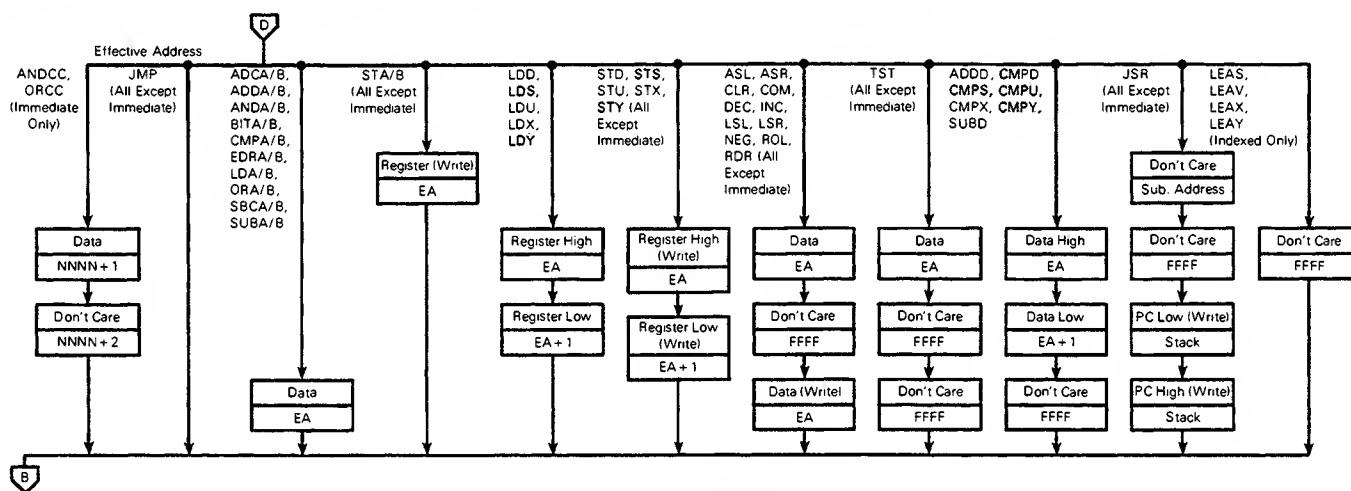


FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 5 of 5)

Constant Offset from R

No Offset
5-Bit Offset
8-Bit Offset
16-Bit Offset

Accumulator Offset from R

A Register Offset
B Register Offset
D Register Offset

Auto Increment/Decrement R

Increment by 1
Increment by 2
Decrement by 1
Decrement by 2

Constant Offset from PC

8-Bit Offset
16-Bit Offset

DirectExtendedImmediate

* The index register is incremented following the indexed access.

Effective Address (EA)

Index Register
Index Register
Index Register + Post Byte
Index Register + Post Byte High, Post Byte Low

Index Register + A Register
Index Register + B Register
Index Register + D Register

Index Register*
Index Register
Index Register - 1
Index Register - 2

Program Counter + Offset Byte
Program Counter + Offset High Byte, Offset Low Byte

Direct Page Register: Address Low

Address High: Address Low

NNNN + 1



TABLE 4 — 8-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

| Mnemonic(s) | Operation |
|-----------------|--|
| ADCA, ADCB | Add memory to accumulator with carry |
| ADDA, ADDB | Add memory to accumulator |
| ANDA, ANDB | And memory with accumulator |
| ASL, ASLA, ASLB | Arithmetic shift of accumulator or memory left |
| ASR, ASRA, ASRB | Arithmetic shift of accumulator or memory right |
| BITA, BITB | Bit test memory with accumulator |
| CLR, CLRA, CLRB | Clear accumulator or memory location |
| CMPA, CMPB | Compare memory from accumulator |
| COM, COMA, COMB | Complement accumulator or memory location |
| DAA | Decimal adjust A accumulator |
| DEC, DECA, DECB | Decrement accumulator or memory location |
| EORA, EORB | Exclusive or memory with accumulator |
| EXG R1, R2 | Exchange R1 with R2 (R1, R2 = A, B, CC, DP) |
| INC, INCA, INCB | Increment accumulator or memory location |
| LDA, LDB | Load accumulator from memory |
| LSL, LSLA, LSLB | Logical shift left accumulator or memory location |
| LSR, LSRA, LSRB | Logical shift right accumulator or memory location |
| MUL | Unsigned multiply ($A \times B \rightarrow D$) |
| NEG, NEGA, NEGB | Negate accumulator or memory |
| ORA, ORB | Or memory with accumulator |
| ROL, ROLA, ROLB | Rotate accumulator or memory left |
| ROR, RORA, RORB | Rotate accumulator or memory right |
| SBCA, SBCB | Subtract memory from accumulator with borrow |
| STA, STB | Store accumulator to memory |
| SUBA, SUBB | Subtract memory from accumulator |
| TST, TSTA, TSTB | Test accumulator or memory location |
| TFR R1, R2 | Transfer R1 to R2 (R1, R2 = A, B, CC, DP) |

NOTE: A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

TABLE 5 — 16-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

| Mnemonic(s) | Operation |
|-------------|--|
| ADDD | Add memory to D accumulator |
| CMPD | Compare memory from D accumulator |
| EXG D, R | Exchange D with X, Y, S, U or PC |
| LDD | Load D accumulator from memory |
| SEX | Sign Extend B accumulator into A accumulator |
| STD | Store D accumulator to memory |
| SUBD | Subtract memory from D accumulator |
| TFR D, R | Transfer D to X, Y, S, U or PC |
| TFR R, D | Transfer X, Y, S, U or PC to D |

NOTE: D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

TABLE 6 — INDEX REGISTER/STACK POINTER INSTRUCTIONS

| Instruction | Description |
|-------------|--|
| CMPS, CMPU | Compare memory from stack pointer |
| CMPX, CMPY | Compare memory from index register |
| EXG R1, R2 | Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC |
| LEAS, LEAU | Load effective address into stack pointer |
| LEAX, LEAY | Load effective address into index register |
| LDS, LDU | Load stack pointer from memory |
| LDX, LDY | Load index register from memory |
| PSHS | Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack |
| PSHU | Push A, B, CC, DP, D, X, Y, S, or PC onto user stack |
| PULS | Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack |
| PULU | Pull A, B, CC, DP, D, X, Y, S or PC from hardware stack |
| STS, STU | Store stack pointer to memory |
| STX, STY | Store index register to memory |
| TFR R1, R2 | Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC |
| ABX | Add B accumulator to X (unsigned) |



TABLE 7 — BRANCH INSTRUCTIONS

| Instruction | Description |
|--------------------------|--|
| SIMPLE BRANCHES | |
| BEQ, LBEQ | Branch if equal |
| BNE, LBNE | Branch if not equal |
| BMI, LBMI | Branch if minus |
| BPL, LBPL | Branch if plus |
| BCS, LBSC | Branch if carry set |
| BCC, LBCC | Branch if carry clear |
| BVS, LBVS | Branch if overflow set |
| BVC, LBVC | Branch if overflow clear |
| SIGNED BRANCHES | |
| BGT, LBGT | Branch if greater (signed) |
| BVS, LBVS | Branch if invalid 2's complement result |
| BGE, LBGE | Branch if greater than or equal (signed) |
| BEQ, LBEQ | Branch if equal |
| BNE, LBNE | Branch if not equal |
| BLE, LBLE | Branch if less than or equal (signed) |
| BVC, LBVC | Branch if valid 2's complement result |
| BLT, LBLT | Branch if less than (signed) |
| UNSIGNED BRANCHES | |
| BHI, LBHI | Branch if higher (unsigned) |
| BCC, LBCC | Branch if higher or same (unsigned) |
| BHS, LBHS | Branch if higher or same (unsigned) |
| BEQ, LBEQ | Branch if equal |
| BNE, LBNE | Branch if not equal |
| BLS, LBLS | Branch if lower or same (unsigned) |
| BCS, LBSC | Branch if lower (unsigned) |
| BLO, LBLO | Branch if lower (unsigned) |
| OTHER BRANCHES | |
| BSR, LBSR | Branch to subroutine |
| BRA, LBRA | Branch always |
| BRN, LBRN | Branch never |

TABLE 8 — MISCELLANEOUS INSTRUCTIONS

| Instruction | Description |
|-----------------|--|
| ANDCC | AND condition code register |
| CWAI | AND condition code register, then wait for interrupt |
| NOP | No operation |
| ORCC | OR condition code register |
| JMP | Jump |
| JSR | Jump to subroutine |
| RTI | Return from interrupt |
| RTS | Return from subroutine |
| SWI, SWI2, SWI3 | Software interrupt (absolute indirect) |
| SYNC | Synchronize with interrupt line |



TABLE 9 — HEXADECIMAL VALUES OF MACHINE CODES

| OP | Mnem | Mode | ~ | # | OP | Mnem | Mode | ~ | # | OP | Mnem | Mode | ~ | # |
|----|----------|----------|----|---|----|------------|----------|------|----|----|----------|----------|----|----|
| 00 | NEG | Direct | 6 | 2 | 30 | LEAX | Indexed | 4+ | 2+ | 60 | NEG | Indexed | 6+ | 2+ |
| 01 | * | ↑ | | | 31 | LEAY | ↑ | 4+ | 2+ | 61 | * | ↑ | | |
| 02 | * | | | | 32 | LEAS | ↓ | 4+ | 2+ | 62 | * | | | |
| 03 | COM | | 6 | 2 | 33 | LEAU | Indexed | 4+ | 2+ | 63 | COM | | 6+ | 2+ |
| 04 | LSR | | 6 | 2 | 34 | PSHS | Immed | 5+ | 2 | 64 | LSR | | 6+ | 2+ |
| 05 | * | | | | 35 | PULS | Immed | 5+ | 2 | 65 | * | | | |
| 06 | ROR | | 6 | 2 | 36 | PSHU | Immed | 5+ | 2 | 66 | ROR | | 6+ | 2+ |
| 07 | ASR | | 6 | 2 | 37 | PULU | Immed | 5+ | 2 | 67 | ASR | | 6+ | 2+ |
| 08 | ASL, LSL | | 6 | 2 | 38 | * | — | | | 68 | ASL, LSL | | 6+ | 2+ |
| 09 | ROL | | 6 | 2 | 39 | RTS | Inherent | 5 | 1 | 69 | ROL | | 6+ | 2+ |
| 0A | DEC | | 6 | 2 | 3A | ABX | ↑ | 3 | 1 | 6A | DEC | | 6+ | 2+ |
| 0B | * | | | | 3B | RTI | ↑ | 6/15 | 1 | 6B | * | | | |
| 0C | INC | | 6 | 2 | 3C | CWAI | ↓ | ≥20 | 2 | 6C | INC | | 6+ | 2+ |
| 0D | TST | | 6 | 2 | 3D | MUL | Inherent | 11 | 1 | 6D | TST | | 6+ | 2+ |
| 0E | JMP | ↓ | 3 | 2 | 3E | * | — | | | 6E | JMP | ↓ | 3+ | 2+ |
| 0F | CLR | Direct | 6 | 2 | 3F | SWI | Inherent | 19 | 1 | 6F | CLR | Indexed | 6+ | 2+ |
| 10 | Page 2 | — | — | — | 40 | NEGA | Inherent | 2 | 1 | 70 | NEG | Extended | 7 | 3 |
| 11 | Page 3 | — | — | — | 41 | * | ↑ | | | 71 | * | ↑ | | |
| 12 | NOP | Inherent | 2 | 1 | 42 | * | | | | 72 | * | | | |
| 13 | SYNC | Inherent | ≥4 | 1 | 43 | COMA | | 2 | 1 | 73 | COM | | 7 | 3 |
| 14 | * | | | | 44 | LSRA | | 2 | 1 | 74 | LSR | | 7 | 3 |
| 15 | * | | | | 45 | * | | | | 75 | * | | | |
| 16 | LBRA | Relative | 5 | 3 | 46 | RORA | | 2 | 1 | 76 | ROR | | 7 | 3 |
| 17 | LBSR | Relative | 9 | 3 | 47 | ASRA | | 2 | 1 | 77 | ASR | | 7 | 3 |
| 18 | * | | | | 48 | ASLA, LSLA | | 2 | 1 | 78 | ASL, LSL | | 7 | 3 |
| 19 | DAA | Inherent | 2 | 1 | 49 | ROLA | | 2 | 1 | 79 | ROL | | 7 | 3 |
| 1A | ORCC | Immed | 3 | 2 | 4A | DECA | | 2 | 1 | 7A | DEC | | 7 | 3 |
| 1B | * | — | | | 4B | * | | | | 7B | * | | | |
| 1C | ANDCC | Immed | 3 | 2 | 4C | INCA | | 2 | 1 | 7C | INC | | 7 | 3 |
| 1D | SEX | Inherent | 2 | 1 | 4D | TSTA | | 2 | 1 | 7D | TST | | 7 | 3 |
| 1E | EXG | Immed | 8 | 2 | 4E | * | ↓ | | | 7E | JMP | ↓ | 4 | 3 |
| 1F | TFR | Immed | 6 | 2 | 4F | CLRA | Inherent | 2 | 1 | 7F | CLR | Extended | 7 | 3 |
| 20 | BRA | Relative | 3 | 2 | 50 | NEGB | Inherent | 2 | 1 | 80 | SUBA | Immed | 2 | 2 |
| 21 | BRN | ↑ | 3 | 2 | 51 | * | ↑ | | | 81 | CMPA | ↑ | 2 | 2 |
| 22 | BHI | | 3 | 2 | 52 | * | | | | 82 | SBCA | | 2 | 2 |
| 23 | BLS | | 3 | 2 | 53 | COMB | | 2 | 1 | 83 | SUBD | | 4 | 3 |
| 24 | BHS, BCC | | 3 | 2 | 54 | LSRB | | 2 | 1 | 84 | ANDA | | 2 | 2 |
| 25 | BLO, BCS | | 3 | 2 | 55 | * | | | | 85 | BITA | | 2 | 2 |
| 26 | BNE | | 3 | 2 | 56 | RORB | | 2 | 1 | 86 | LDA | | 2 | 2 |
| 27 | BEQ | | 3 | 2 | 57 | ASRB | | 2 | 1 | 87 | * | | | |
| 28 | BVC | | 3 | 2 | 58 | ASLB, LSLB | | 2 | 1 | 88 | EORA | | 2 | 2 |
| 29 | BVS | | 3 | 2 | 59 | ROLB | | 2 | 1 | 89 | ADCA | | 2 | 2 |
| 2A | BPL | | 3 | 2 | 5A | DECB | | 2 | 1 | 8A | ORA | | 2 | 2 |
| 2B | BMI | | 3 | 2 | 5B | * | | | | 8B | ADDA | | 2 | 2 |
| 2C | BGE | | 3 | 2 | 5C | INCB | | 2 | 1 | 8C | CMPX | Immed | 4 | 3 |
| 2D | BLT | ↓ | 3 | 2 | 5D | TSTB | | 2 | 1 | 8D | BSR | Relative | 7 | 2 |
| 2E | BGT | | 3 | 2 | 5E | * | ↓ | | | 8E | LDX | Immed | 3 | 3 |
| 2F | BLE | Relative | 3 | 2 | 5F | CLRB | Inherent | 2 | 1 | 8F | * | | | |

LEGEND:

~ Number of MPU cycles (less possible push pull or indexed-mode cycles)

Number of program bytes

* Denotes unused opcode

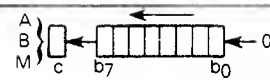
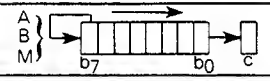


TABLE 9 — HEXADECIMAL VALUES OF MACHINE CODES (CONTINUED)

| OP | Mnem | Mode | ~ | # | OP | Mnem | Mode | ~ | # | OP | Mnem | Mode | ~ | # |
|---|------|----------|----|----|----|------|----------|----|----|---------------------------------------|------------|----------|------|----|
| 90 | SUBA | Direct | 4 | 2 | C0 | SUBB | Immed | 2 | 2 | <div>Page 2 and 3 Machine Codes</div> | | | | |
| 91 | CMPA | ↑ | 4 | 2 | C1 | CMPB | ↑ | 2 | 2 | | | | | |
| 92 | SBCA | ↑ | 4 | 2 | C2 | SBCB | ↑ | 2 | 2 | | | | | |
| 93 | SUBD | ↑ | 6 | 2 | C3 | ADDD | ↑ | 4 | 3 | | | | | |
| 94 | ANDA | ↑ | 4 | 2 | C4 | ANDB | ↑ | 2 | 2 | | | | | |
| 95 | BITA | ↑ | 4 | 2 | C5 | BITB | ↑ | 2 | 2 | | | | | |
| 96 | LDA | ↑ | 4 | 2 | C6 | LDB | ↑ | 2 | 2 | | | | | |
| 97 | STA | ↑ | 4 | 2 | C7 | * | ↑ | 2 | 2 | | | | | |
| 98 | EORA | ↑ | 4 | 2 | C8 | EORB | ↑ | 2 | 2 | | | | | |
| 99 | ADCA | ↑ | 4 | 2 | C9 | ADCB | ↑ | 2 | 2 | | | | | |
| 9A | ORA | ↑ | 4 | 2 | CA | ORB | ↑ | 2 | 2 | 1021 | LBRN | Relative | 5 | 4 |
| 9B | ADDA | ↑ | 4 | 2 | CB | ADDB | ↑ | 2 | 2 | 1022 | LBHI | ↑ | 5(6) | 4 |
| 9C | CMPX | ↑ | 6 | 2 | CC | LDD | ↑ | 3 | 3 | 1023 | LBSL | ↑ | 5(6) | 4 |
| 9D | JSR | ↑ | 7 | 2 | CD | * | ↑ | 3 | 3 | 1024 | LBHS, LBCC | ↑ | 5(6) | 4 |
| 9E | LDX | ↑ | 5 | 2 | CE | LDU | ↑ | 3 | 3 | 1025 | LBCS, LBLO | ↑ | 5(6) | 4 |
| 9F | STX | Direct | 5 | 2 | CF | * | Immed | 3 | 3 | 1026 | LBNE | ↑ | 5(6) | 4 |
| A0 | SUBA | Indexed | 4+ | 2+ | D0 | SUBB | Direct | 4 | 2 | 1027 | LBEQ | ↑ | 5(6) | 4 |
| A1 | CMPA | ↑ | 4+ | 2+ | D1 | CMPB | ↑ | 4 | 2 | 1028 | LBVC | ↑ | 5(6) | 4 |
| A2 | SBCA | ↑ | 4+ | 2+ | D2 | SBCB | ↑ | 4 | 2 | 1029 | LBVS | ↑ | 5(6) | 4 |
| A3 | SUBD | ↑ | 6+ | 2+ | D3 | ADDD | ↑ | 6 | 2 | 102A | LBPL | ↑ | 5(6) | 4 |
| A4 | ANDA | ↑ | 4+ | 2+ | D4 | ANDB | ↑ | 4 | 2 | 102B | LBMI | ↑ | 5(6) | 4 |
| A5 | BITA | ↑ | 4+ | 2+ | D5 | BITB | ↑ | 4 | 2 | 102C | LBGE | ↑ | 5(6) | 4 |
| A6 | LDA | ↑ | 4+ | 2+ | D6 | LDB | ↑ | 4 | 2 | 102D | LBLT | ↑ | 5(6) | 4 |
| A7 | STA | ↑ | 4+ | 2+ | D7 | STB | ↑ | 4 | 2 | 102E | LBGT | ↑ | 5(6) | 4 |
| A8 | EORA | ↑ | 4+ | 2+ | D8 | EORB | ↑ | 4 | 2 | 102F | LBLE | ↑ | 5(6) | 4 |
| A9 | ADCA | ↑ | 4+ | 2+ | D9 | ADCB | ↑ | 4 | 2 | 103F | SWI2 | Relative | 20 | 2 |
| AA | ORA | ↑ | 4+ | 2+ | DA | ORB | ↑ | 4 | 2 | 1083 | CMPD | Inherent | 5 | 4 |
| AB | ADDA | ↑ | 4+ | 2+ | DB | ADDB | ↑ | 4 | 2 | 108C | CMPY | Immed | 5 | 4 |
| AC | CMPX | ↑ | 6+ | 2+ | DC | LDD | ↑ | 5 | 2 | 108E | LDY | Immed | 4 | 4 |
| AD | JSR | ↑ | 7+ | 2+ | DD | STD | ↑ | 5 | 2 | 1093 | CMPD | Direct | 7 | 3 |
| AE | LDX | ↑ | 5+ | 2+ | DE | LDU | ↑ | 5 | 2 | 109C | CMPY | ↑ | 7 | 3 |
| AF | STX | Indexed | 5+ | 2+ | DF | STU | Direct | 5 | 2 | 109E | LDY | ↓ | 6 | 3 |
| B0 | SUBA | Extended | 5 | 3 | E0 | SUBB | Indexed | 4+ | 2+ | 109F | STY | Direct | 6 | 3 |
| B1 | CMPA | ↑ | 5 | 3 | E1 | CMPB | ↑ | 4+ | 2+ | 10A3 | CMPD | Indexed | 7+ | 3+ |
| B2 | SBCA | ↑ | 5 | 3 | E2 | SBCB | ↑ | 4+ | 2+ | 10AC | CMPY | ↑ | 7+ | 3+ |
| B3 | SUBD | ↑ | 7 | 3 | E3 | ADDD | ↑ | 6+ | 2+ | 10AE | LDY | ↑ | 6+ | 3+ |
| B4 | ANDA | ↑ | 5 | 3 | E4 | ANDB | ↑ | 4+ | 2+ | 10AF | STY | ↑ | 6+ | 3+ |
| B5 | BITA | ↑ | 5 | 3 | E5 | BITB | ↑ | 4+ | 2+ | 10B3 | CMPD | Indexed | 8 | 4 |
| B6 | LDA | ↑ | 5 | 3 | E6 | LDB | ↑ | 4+ | 2+ | 10BC | CMPY | ↑ | 8 | 4 |
| B7 | STA | ↑ | 5 | 3 | E7 | STB | ↑ | 4+ | 2+ | 10BE | LDY | ↑ | 7 | 4 |
| BB | EORA | ↑ | 5 | 3 | EB | EORB | ↑ | 4+ | 2+ | 10BF | STY | Extended | 7 | 4 |
| B9 | ADCA | ↑ | 5 | 3 | E9 | ADCB | ↑ | 4+ | 2+ | 10CE | LDS | Immed | 4 | 4 |
| BA | ORA | ↑ | 5 | 3 | EA | ORB | ↑ | 4+ | 2+ | 10DE | LDS | Direct | 6 | 3 |
| BB | ADDA | ↑ | 5 | 3 | EB | ADDB | ↑ | 4+ | 2+ | 10DF | STS | Direct | 6 | 3 |
| BC | CMPX | ↑ | 7 | 3 | EC | LDD | ↑ | 5+ | 2+ | 10EE | LDS | Indexed | 6+ | 3+ |
| BD | JSR | ↑ | 8 | 3 | ED | STD | ↑ | 5+ | 2+ | 10EF | STS | Indexed | 6+ | 3+ |
| BE | LDX | ↑ | 6 | 3 | EE | LDU | ↑ | 5+ | 2+ | 10FE | LDS | Extended | 7 | 4 |
| BF | STX | Extended | 6 | 3 | EF | STU | Indexed | 5+ | 2+ | 10FF | STS | Extended | 7 | 4 |
| NOTE: All unused opcodes are both undefined and illegal | | | | | F0 | SUBB | Extended | 5 | 3 | 113F | SWI3 | Inherent | 20 | 2 |
| | | | | | F1 | CMPB | ↑ | 5 | 3 | 1183 | CMPU | Immed | 5 | 4 |
| | | | | | F2 | SBCB | ↑ | 5 | 3 | 118C | CMPS | Immed | 5 | 4 |
| | | | | | F3 | ADDD | ↑ | 7 | 3 | 1193 | CMPU | Direct | 7 | 3 |
| | | | | | F4 | ANDB | ↑ | 5 | 3 | 119C | CMPS | Direct | 7 | 3 |
| | | | | | F5 | BITB | ↑ | 5 | 3 | 11A3 | CMPU | Indexed | 7+ | 3+ |
| | | | | | F6 | LDB | ↑ | 5 | 3 | 11AC | CMPS | Indexed | 7+ | 3+ |
| | | | | | F7 | STB | ↑ | 5 | 3 | 11B3 | CMPU | Extended | 8 | 4 |
| | | | | | FB | EORB | ↑ | 5 | 3 | 11BC | CMPS | Extended | 8 | 4 |
| | | | | | F9 | ADCB | ↑ | 5 | 3 | | | | | |
| | | | | | FA | ORB | ↑ | 5 | 3 | | | | | |
| | | | | | FB | ADDB | Extended | 5 | 3 | | | | | |
| | | | | | FC | LDD | Extended | 6 | 3 | | | | | |
| | | | | | FD | STD | ↑ | 6 | 3 | | | | | |
| | | | | | FE | LDU | ↑ | 6 | 3 | | | | | |
| | | | | | FF | STU | Extended | 6 | 3 | | | | | |



FIGURE 18 — PROGRAMMING AID

| Instruction | Forms | Addressing Modes | | | | | | | | | | | | | | | Description | 5 H | 3 N | 2 Z | 1 V | 0 C | |
|-------------|---|--|---------------------------------|---------------------------------|--|------------------------------------|---------------------------------|--|--|--|--|---------------------------------|---------------------------------|----------|--------|--------|---|---------------------------------|--------|--------|--------|--------|---|
| | | Immediate | | | Direct | | | Indexed | | | Extended | | | Inherent | | | | | | | | | |
| | | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | | | | | | | |
| ABX | | | | | | | | | | | | | | 3A | 3 | 1 | B + X ← X (Unsigned) | • | • | • | • | • | |
| ADC | ADCA ADCB | 89 C9 | 2 2 | 2 2 | 99 D9 | 4 4 | 2 2 | A9 E9 | 4+ 4+ | 2+ 2+ | B9 F9 | 5 5 | 3 3 | | | | A + M + C ← A B + M + C ← B | † | † | † | † | † | |
| ADD | ADDA ADDB ADDD | 8B CB C3 | 2 2 4 | 2 2 3 | 9B DB D3 | 4 4 6 | 2 2 2 | AB EB E3 | 4+ 4+ 6+ | 2+ 2+ 2+ | BB FB F3 | 5 5 7 | 3 3 3 | | | | A + M ← A B + M ← B D + M: M + 1 ← D | † | † | † | † | † | |
| AND | ANDA ANDB ANDCC | 84 C4 1C | 2 2 3 | 2 2 2 | 94 D4 | 4 4 | 2 2 | A4 E4 | 4+ 4+ | 2+ 2+ | B4 F4 | 5 5 | 3 3 | | | | A ∧ M ← A B ∧ M ← B CC ∧ IMM ← CC | • | † | † | 0 | • | |
| ASL | ASLA ASL8 ASL | | | | | | | | | | | | | 48 58 | 2 2 | 1 1 |  | 8 8 8 | † | † | † | † | † |
| ASR | ASRA ASR8 ASR | | | | | | | | | | | | | 47 57 | 2 2 | 1 1 |  | 8 8 8 | † | † | • | † | † |
| BIT | BITA BITB | 85 C5 | 2 2 | 2 2 | 95 D5 | 4 4 | 2 2 | A5 E5 | 4+ 4+ | 2+ 2+ | B5 F5 | 5 5 | 3 3 | | | | Bit Test A (M ∧ A) Bit Test B (M ∧ B) | • | † | † | 0 | • | |
| CLR | CLRA CLRB CLR | | | | | | | | | | | | | 4F 5F | 2 2 | 1 1 | 0 ← A 0 ← B 0 ← M | • | 0 | † | 0 | 0 | |
| CMP | CMPS CMPB CMPD CMPU CMPX CMPY | 81 C1 10 83 11 8C 83 | 2 2 5 5 5 5 5 | 2 2 4 4 4 4 4 | 91 D1 10 93 11 9C 93 | 4 4 7 A3 7 AC A3 | 2 2 3 3 3 3 3 | A1 E1 10 A3 11 AC A3 | 4+ 4+ 7+ 7+ 7+ 7+ 7+ | 2+ 2+ 3+ 3+ 3+ 3+ 3+ | B1 F1 10 B3 11 BC B3 | 5 5 8 8 8 8 8 | 3 3 4 4 4 4 4 | | | | Compare M from A Compare M from B Compare M: M + 1 from D Compare M: M + 1 from S Compare M: M + 1 from U Compare M: M + 1 from X Compare M: M + 1 from Y | 8 8 • • • • • | † | † | † | † | † |
| COM | COMA COMB COM | | | | | | | | | | | | | 43 53 | 2 2 | 1 1 | A ← A B ← B M ← M | • | † | † | 0 | 1 | |
| CWAI | | 3C | ≥20 | 2 | | | | | | | | | | | | | CC ∧ IMM ← CC Wait for Interrupt | | | | | 7 | |
| DAA | | | | | | | | | | | | | | 19 | 2 | 1 | Decimal Adjust A | • | † | † | 0 | † | |
| DEC | DECA DECB DEC | | | | | | | | | | | | | 4A 5A | 2 2 | 1 1 | A ← A - 1 B ← B - 1 M ← M - 1 | • | † | † | † | • | |
| EOR | EORA EORB | 88 C8 | 2 2 | 2 2 | 98 D8 | 4 4 | 2 2 | A8 E8 | 4+ 4+ | 2+ 2+ | B8 F8 | 5 5 | 3 3 | | | | A ⊕ M ← A B ⊕ M ← B | • | † | † | 0 | • | |
| EXG | R1, R2 | 1E | 8 | 2 | | | | | | | | | | | | | R1 ← R2 ² | • | • | • | • | • | |
| INC | INCA INCB INC | | | | | | | | | | | | | 4C 5C | 2 2 | 1 1 | A + 1 ← A B + 1 ← B M + 1 ← M | • | † | † | † | • | |
| JMP | | | | | 0E | 3 | 2 | 6E | 3+ | 2+ | 7E | 4 | 3 | | | | EA ³ ← PC | • | • | • | • | • | |
| JSR | | | | | 9D | 7 | 2 | AD | 7+ | 2+ | BD | 8 | 3 | | | | Jump to Subroutine | • | • | • | • | • | |
| LD | LDA LDB LDD LDS LDU LDX LDY | 86 C6 CC 10 CE 8E 10 | 2 2 3 4 3 3 4 | 2 2 3 4 3 3 4 | 96 D6 DC 10 DE 9E 10 | 4 4 5 6 5 5 6 | 2 2 2 3 2 2 3 | A6 E6 EC 10 EE AE 10 | 4+ 4+ 5+ 6+ 5+ 5+ 6+ | 2+ 2+ 2+ 3+ 2+ 2+ 3+ | 86 F6 FC 10 FE BE 10 | 5 5 6 7 6 6 7 | 3 3 3 4 3 3 4 | | | | M ← A M ← B M: M + 1 ← D M: M + 1 ← S M: M + 1 ← U M: M + 1 ← X M: M + 1 ← Y | • | † | † | 0 | • | |
| LEA | LEAS LEAU LEAX LEAY | | | | | | | | | | | | | | | | EA ³ ← S EA ³ ← U EA ³ ← X EA ³ ← Y | • | • | • | • | • | |

LEGEND:

OP Operation Code (Hexadecimal)

~ Number of MPU Cycles

Number of Program Bytes

+ Arithmetic Plus

- Arithmetic Minus

• Multiply

 \bar{M} Complement of M

→ Transfer Into

H Half-carry (from bit 3)

N Negative (sign bit)

Z Zero result

V Overflow, 2's complement

C Carry from ALU

† Test and set if true, cleared otherwise

• Not Affected

CC Condition Code Register

: Concatenation

V Logical or

Λ Logical and

⊕ Logical Exclusive or



MOTOROLA Semiconductor Products Inc.

FIGURE 18 — PROGRAMMING AID (CONTINUED)

| Instruction | Forms | Addressing Modes | | | | | | | | | | | | | | | Description | 5 | 3 | 2 | 1 | 0 | | |
|-------------|-------------------|------------------|-----|---|--------|---|---|----------------------|----|----|----------|---|----|----------|---|---|--|---|---|---|---|---|---|---|
| | | Immediate | | | Direct | | | Indexed ¹ | | | Extended | | | Inherent | | | | H | N | Z | V | C | | |
| | | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | | | | | | | | |
| LSL | LSLA | | | | | | | | | | | | 48 | 2 | 1 | | • | • | • | • | • | | | |
| | LSLB | | | | | | | | | | | | 58 | 2 | 1 | | • | • | • | • | • | | | |
| | LSL | | | | 08 | 6 | 2 | 68 | 6+ | 2+ | 78 | 7 | 3 | | | | | • | • | • | • | • | | |
| LSR | LSRA | | | | | | | | | | | | 44 | 2 | 1 | | • | 0 | • | • | • | | | |
| | LSRB | | | | | | | | | | | | 54 | 2 | 1 | | • | 0 | • | • | • | | | |
| | LSR | | | | 04 | 6 | 2 | 64 | 6+ | 2+ | 74 | 7 | 3 | | | | | • | 0 | • | • | • | | |
| MUL | | | | | | | | | | | | | 3D | 11 | 1 | A × B ← D (Unsigned) | | | • | • | • | • | 9 | |
| NEG | NEGA | | | | | | | | | | | | 40 | 2 | 1 | $\bar{A} + 1 - A$ $\bar{B} + 1 - B$ $\bar{M} + 1 - M$ | 8 | • | • | • | • | | | |
| | NEGB | | | | | | | | | | | | 50 | 2 | 1 | | 8 | • | • | • | • | | | |
| | NEG | | | | 00 | 6 | 2 | 60 | 6+ | 2+ | 70 | 7 | 3 | | | | | 8 | • | • | • | • | | |
| NOP | | | | | | | | | | | | | 12 | 2 | 1 | No Operation | | | • | • | • | • | • | |
| OR | ORA | 8A | 2 | 2 | 9A | 4 | 2 | AA | 4+ | 2+ | BA | 5 | 3 | | | | $A \vee M \leftarrow A$ $B \vee M \leftarrow B$ $CC \vee IMM \leftarrow CC$ | • | • | • | 0 | • | | |
| | ORB | CA | 2 | 2 | DA | 4 | 2 | EA | 4+ | 2+ | FA | 5 | 3 | | | | | • | • | • | 0 | • | | |
| | ORCC | 1A | 3 | 2 | | | | | | | | | | | | | | • | • | • | • | 7 | | |
| PSH | PSHS | 34 | 5+4 | 2 | | | | | | | | | | | | | Push Registers on S Stack | | | • | • | • | • | • |
| | PSHU | 36 | 5+4 | 2 | | | | | | | | | | | | | Push Registers on U Stack | | | • | • | • | • | • |
| PUL | PULS | 35 | 5+4 | 2 | | | | | | | | | | | | | Pull Registers from S Stack | | | • | • | • | • | • |
| | PULU | 37 | 5+4 | 2 | | | | | | | | | | | | | Pull Registers from U Stack | | | • | • | • | • | • |
| ROL | ROLA | | | | | | | | | | | | 49 | 2 | 1 | | • | • | • | • | • | | | |
| | ROLB | | | | | | | | | | | | 59 | 2 | 1 | | • | • | • | • | • | | | |
| | ROL | | | | 09 | 6 | 2 | 69 | 6+ | 2+ | 79 | 7 | 3 | | | | | • | • | • | • | • | | |
| ROR | RORA | | | | | | | | | | | | 46 | 2 | 1 | | • | • | • | • | • | | | |
| | RORB | | | | | | | | | | | | 56 | 2 | 1 | | • | • | • | • | • | | | |
| | ROR | | | | 06 | 6 | 2 | 66 | 6+ | 2+ | 76 | 7 | 3 | | | | | • | • | • | • | • | | |
| RTI | | | | | | | | | | | | | 3B | 6/15 | 1 | Return From Interrupt | | | | | | | 7 | |
| RTS | | | | | | | | | | | | | 39 | 5 | 1 | Return from Subroutine | | | • | • | • | • | • | |
| SBC | SBCA | 82 | 2 | 2 | 92 | 4 | 2 | A2 | 4+ | 2+ | B2 | 5 | 3 | | | | $A - M - C \leftarrow A$ $B - M - C \leftarrow B$ | 8 | • | • | • | • | | |
| | SBCB | C2 | 2 | 2 | D2 | 4 | 2 | E2 | 4+ | 2+ | F2 | 5 | 3 | | | | | 8 | • | • | • | • | | |
| SEX | | | | | | | | | | | | | 1D | 2 | 1 | Sign Extend B into A | | | • | • | • | 0 | • | |
| ST | STA | | | | 97 | 4 | 2 | A7 | 4+ | 2+ | B7 | 5 | 3 | | | | $A \leftarrow M$ $B \leftarrow M$ $D \leftarrow M:M+1$ $S \leftarrow M:M+1$ | • | • | • | 0 | • | | |
| | STB | | | | D7 | 4 | 2 | E7 | 4+ | 2+ | F7 | 5 | 3 | | | | | • | • | • | 0 | • | | |
| | STD | | | | DD | 5 | 2 | ED | 5+ | 2+ | FD | 6 | 3 | | | | | • | • | • | 0 | • | | |
| | STS | | | | 10 | 6 | 3 | 10 | 6+ | 3+ | 10 | 7 | 4 | | | | | • | • | • | 0 | • | | |
| | | | | | DF | 6 | 3 | EF | 6+ | 3+ | FF | 7 | 4 | | | | $U \leftarrow M:M+1$ $X \leftarrow M:M+1$ $Y \leftarrow M:M+1$ | • | • | • | 0 | • | | |
| | STU | | | | DF | 5 | 2 | EF | 5+ | 2+ | FF | 6 | 3 | | | | | • | • | • | 0 | • | | |
| | STX | | | | 9F | 5 | 2 | AF | 5+ | 2+ | BF | 6 | 3 | | | | | • | • | • | 0 | • | | |
| | STY | | | | 10 | 6 | 3 | 10 | 6+ | 3+ | 10 | 7 | 4 | | | | | • | • | • | 0 | • | | |
| | | | | | 9F | 6 | 3 | AF | 6+ | 3+ | BF | 7 | 4 | | | | | • | • | • | 0 | • | | |
| | | | | | | | | | | | | | | | | | | • | • | • | 0 | • | | |
| SUB | SUBA | 80 | 2 | 2 | 90 | 4 | 2 | A0 | 4+ | 2+ | B0 | 5 | 3 | | | | $A - M \leftarrow A$ $B - M \leftarrow B$ $D - M:M+1 \leftarrow D$ | 8 | • | • | • | • | | |
| | SUBB | C0 | 2 | 2 | D0 | 4 | 2 | E0 | 4+ | 2+ | F0 | 5 | 3 | | | | | 8 | • | • | • | • | | |
| | SUBD | 83 | 4 | 3 | 93 | 6 | 2 | A3 | 6+ | 2+ | B3 | 7 | 3 | | | | | • | • | • | • | • | | |
| SWI | SWI ⁶ | | | | | | | | | | | | 3F | 19 | 1 | Software Interrupt 1 | • | • | • | • | • | | | |
| | SWI ²⁶ | | | | | | | | | | | | 10 | 20 | 2 | | • | • | • | • | • | | | |
| | | | | | | | | | | | | | 3F | | | | Software Interrupt 3 | • | • | • | • | • | | |
| | SWI ³⁶ | | | | | | | | | | | | 11 | 20 | 1 | | | • | • | • | • | • | | |
| SYNC | | | | | | | | | | | | | 13 | ≥4 | 1 | Synchronize to Interrupt | | | • | • | • | • | • | |
| TFR | R1, R2 | 1F | 6 | 2 | | | | | | | | | | | | $R1 \leftarrow R2^2$ | | | • | • | • | • | • | |
| TST | TSTA | | | | | | | | | | | | 4D | 2 | 1 | Test A Test B Test M | • | • | • | 0 | • | | | |
| | TSTB | | | | | | | | | | | | 5D | 2 | 1 | | • | • | • | 0 | • | | | |
| | TST | | | | 0D | 6 | 2 | 6D | 6+ | 2+ | 7D | 7 | 3 | | | | | • | • | • | 0 | • | | |

NOTES:

- This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table, Table 2.
- R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC
- EA is the effective address.
- The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.
- 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).
- SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
- Conditions Codes set as a direct result of the instruction.
- Value of half-carry flag is undefined.
- Special Case — Carry set if b7 is SET.



FIGURE 18 — PROGRAMMING AID (CONTINUED)

Branch Instructions

| Instruction | Forms | Addressing Mode | | | Description | 5 | 3 | 2 | 1 | 0 |
|-------------|-------|-----------------|------|---|----------------------------|---|---|---|---|---|
| | | OP | ~ | # | | | | | | |
| BCC | BCC | 24 | 3 | 2 | Branch C=0 | • | • | • | • | • |
| | LBCC | 10 | 5(6) | 4 | Long Branch C=0 | • | • | • | • | • |
| | | 24 | | | | | | | | |
| BCS | BCS | 25 | 3 | 2 | Branch C=1 | • | • | • | • | • |
| | LBGS | 10 | 5(6) | 4 | Long Branch C=1 | • | • | • | • | • |
| | | 25 | | | | | | | | |
| BEQ | BEQ | 27 | 3 | 2 | Branch Z=1 | • | • | • | • | • |
| | LBEG | 10 | 5(6) | 4 | Long Branch Z=1 | • | • | • | • | • |
| | | 27 | | | | | | | | |
| BGE | BGE | 2C | 3 | 2 | Branch \geq Zero | • | • | • | • | • |
| | LBGE | 10 | 5(6) | 4 | Long Branch \geq Zero | • | • | • | • | • |
| | | 2C | | | | | | | | |
| BGT | BGT | 2E | 3 | 2 | Branch > Zero | • | • | • | • | • |
| | LBGT | 10 | 5(6) | 4 | Long Branch > Zero | • | • | • | • | • |
| | | 2E | | | | | | | | |
| BHI | BHI | 22 | 3 | 2 | Branch Higher | • | • | • | • | • |
| | LBHI | 10 | 5(6) | 4 | Long Branch Higher | • | • | • | • | • |
| | | 22 | | | | | | | | |
| BHS | BHS | 24 | 3 | 2 | Branch Higher or Same | • | • | • | • | • |
| | LBHS | 10 | 5(6) | 4 | Long Branch Higher or Same | • | • | • | • | • |
| | | 24 | | | | | | | | |
| BLE | BLE | 2F | 3 | 2 | Branch \leq Zero | • | • | • | • | • |
| | LBLE | 10 | 5(6) | 4 | Long Branch \leq Zero | • | • | • | • | • |
| | | 2F | | | | | | | | |
| BLO | BLO | 25 | 3 | 2 | Branch lower | • | • | • | • | • |
| | LBLO | 10 | 5(6) | 4 | Long Branch Lower | • | • | • | • | • |
| | | 25 | | | | | | | | |

| Instruction | Forms | Addressing Mode | | | Description | 5 | 3 | 2 | 1 | 0 |
|-------------|-------|-----------------|------|---|---------------------------|---|---|---|---|---|
| | | OP | ~ | # | | | | | | |
| BLS | BLS | 23 | 3 | 2 | Branch Lower or Same | • | • | • | • | • |
| | LBLS | 10 | 5(6) | 4 | Long Branch Lower or Same | • | • | • | • | • |
| | | 23 | | | | | | | | |
| BLT | BLT | 2D | 3 | 2 | Branch < Zero | • | • | • | • | • |
| | LBLT | 10 | 5(6) | 4 | Long Branch < Zero | • | • | • | • | • |
| | | 2D | | | | | | | | |
| BMI | BMI | 2B | 3 | 2 | Branch Minus | • | • | • | • | • |
| | LBMI | 10 | 5(6) | 4 | Long Branch Minus | • | • | • | • | • |
| | | 2B | | | | | | | | |
| BNE | BNE | 26 | 3 | 2 | Branch Z=0 | • | • | • | • | • |
| | LBNE | 10 | 5(6) | 4 | Long Branch Z=0 | • | • | • | • | • |
| | | 26 | | | | | | | | |
| BPL | BPL | 2A | 3 | 2 | Branch Plus | • | • | • | • | • |
| | LBPL | 10 | 5(6) | 4 | Long Branch Plus | • | • | • | • | • |
| | | 2A | | | | | | | | |
| BRA | BRA | 20 | 3 | 2 | Branch Always | • | • | • | • | • |
| | LBRA | 16 | 5 | 3 | Long Branch Always | • | • | • | • | • |
| | | | | | | | | | | |
| BRN | BRN | 21 | 3 | 2 | Branch Never | • | • | • | • | • |
| | LB RN | 10 | 5 | 4 | Long Branch Never | • | • | • | • | • |
| | | 21 | | | | | | | | |
| BSR | BSR | BD | 7 | 2 | Branch to Subroutine | • | • | • | • | • |
| | LBSR | 17 | 9 | 3 | Long Branch to Subroutine | • | • | • | • | • |
| | | | | | | | | | | |
| BVC | BVC | 28 | 3 | 2 | Branch V=0 | • | • | • | • | • |
| | LBVC | 10 | 5(6) | 4 | Long Branch V=0 | • | • | • | • | • |
| | | 28 | | | | | | | | |
| BVS | BVS | 29 | 3 | 2 | Branch V=1 | • | • | • | • | • |
| | LBVS | 10 | 5(6) | 4 | Long Branch V=1 | • | • | • | • | • |
| | | 29 | | | | | | | | |

SIMPLE BRANCHES

| | OP | ~ | # |
|------|------|---|---|
| BRA | 20 | 3 | 2 |
| LBRA | 16 | 5 | 3 |
| BRN | 21 | 3 | 2 |
| LBRN | 1021 | 5 | 4 |
| BSR | 8D | 7 | 2 |
| LBSR | 17 | 9 | 3 |

SIMPLE CONDITIONAL BRANCHES (Notes 1-4)

| Test | True | OP | False | OP |
|------|------|----|-------|----|
| N=1 | BMI | 2B | BPL | 2A |
| Z=1 | BEQ | 27 | BNE | 26 |
| V=1 | BVS | 29 | BVC | 28 |
| C=1 | BCS | 25 | BCC | 24 |

SIGNED CONDITIONAL BRANCHES (Notes 1-4)

| Test | True | OP | False | OP |
|------------|------|----|-------|----|
| $r > m$ | BGT | 2E | BLE | 2F |
| $r \geq m$ | BGE | 2C | BLT | 2D |
| $r = m$ | BEQ | 27 | BNE | 26 |
| $r \leq m$ | BLE | 2F | BGT | 2E |
| $r < m$ | BLT | 2D | BGE | 2C |

UNSIGNED CONDITIONAL BRANCHES (Notes 1-4)

| Test | True | OP | False | OP |
|------------|------|----|-------|----|
| $r > m$ | BHI | 22 | BLS | 23 |
| $r \geq m$ | BHS | 24 | BLO | 25 |
| $r = m$ | BEQ | 27 | BNE | 26 |
| $r \leq m$ | BLS | 23 | BHI | 22 |
| $r < m$ | BLO | 25 | BHS | 24 |

NOTES:

1. All conditional branches have both short and long variations.
2. All short branches are 2 bytes and require 3 cycles.
3. All conditional long branches are formed by prefixing the short branch opcode with \$10 and using a 16-bit destination offset.
4. All conditional long branches require 4 bytes and 6 cycles if the branch is taken or 5 cycles if the branch is not taken.
5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.



INDEXED ADDRESSING MODES

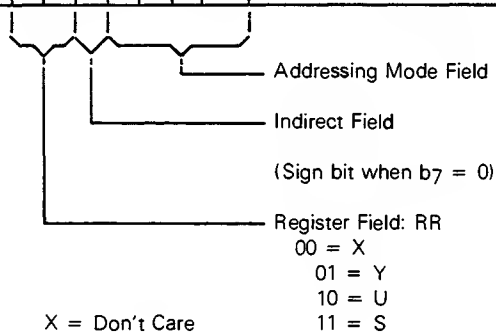
| Type | Forms | Nondirect | | | | Indirect | | | |
|----------------------------|-------------------|----------------|------------------|---|---|-------------------|------------------|---|---|
| | | Assembler Form | Post-Byte Opcode | + | # | Assembler Form | Post-Byte Opcode | + | # |
| Constant Offset From R | No Offset | , R | 1RR00100 | 0 | 0 | [, R] | 1RR10100 | 3 | 0 |
| | 5-Bit Offset | n, R | 0RRnnnnn | 1 | 0 | defaults to 8-bit | | | |
| | 8-Bit Offset | n, R | 1RR01000 | 1 | 1 | [n, R] | 1RR11000 | 4 | 1 |
| | 16-Bit Offset | n, R | 1RR01001 | 4 | 2 | [n, R] | 1RR11001 | 7 | 2 |
| Accumulator Offset From R | A—Register Offset | A, R | 1RR00110 | 1 | 0 | [A, R] | 1RR10110 | 4 | 0 |
| | B—Register Offset | B, R | 1RR00101 | 1 | 0 | [B, R] | 1RR10101 | 4 | 0 |
| | D—Register Offset | D, R | 1RR01011 | 4 | 0 | [D, R] | 1RR11011 | 7 | 0 |
| Auto Increment/Decrement R | Increment By 1 | , R + | 1RR00000 | 2 | 0 | not allowed | | | |
| | Increment By 2 | , R ++ | 1RR00001 | 3 | 0 | [, R ++] | 1RR10001 | 6 | 0 |
| | Decrement By 1 | , -R | 1RR00010 | 2 | 0 | not allowed | | | |
| | Decrement By 2 | , --R | 1RR00011 | 3 | 0 | [, --R] | 1RR10011 | 6 | 0 |
| Constant Offset From PC | 8-Bit Offset | n, PCR | 1XX01100 | 1 | 1 | [n, PCR] | 1XX11100 | 4 | 1 |
| | 16-Bit Offset | n, PCR | 1XX01101 | 5 | 2 | [n, PCR] | 1XX11101 | 8 | 2 |
| Extended Indirect | 16-Bit Address | — | — | — | — | [n] | 10011111 | 5 | 2 |

R = X, Y, U, or S
X = Don't Care

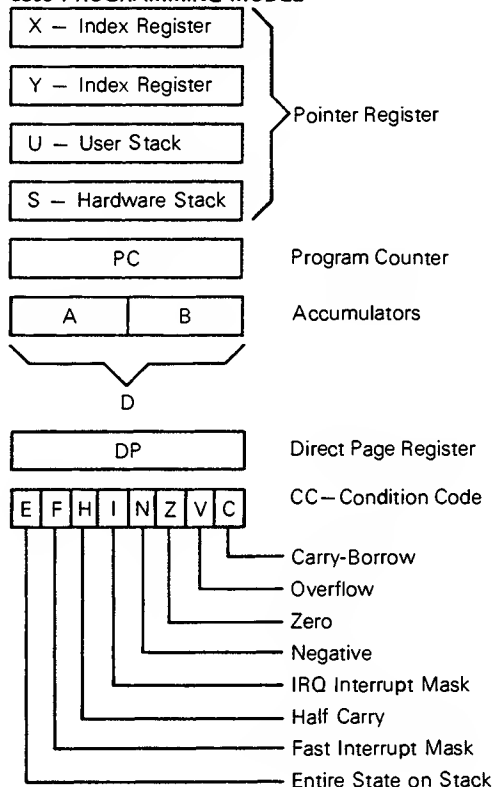
RR: 00 = X 10 = U
01 = Y 11 = S

INDEXED ADDRESSING POSTBYTE
REGISTER BIT ASSIGNMENTS

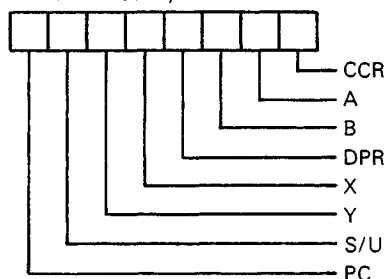
| Post-Byte Register Bit | | | | | | | | Indexed Addressing Mode |
|------------------------|---|---|---|---|---|---|---|---------------------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 0 | R | R | x | x | x | x | x | EA = , R + 5 Bit Offset |
| 1 | R | R | 0 | 0 | 0 | 0 | 0 | , R + |
| 1 | R | R | 1 | 0 | 0 | 0 | 1 | , R ++ |
| 1 | R | R | 0 | 0 | 0 | 1 | 0 | , -R |
| 1 | R | R | 1 | 0 | 0 | 1 | 1 | , --R |
| 1 | R | R | 1 | 0 | 1 | 0 | 0 | EA = , R + 0 Offset |
| 1 | R | R | 1 | 0 | 1 | 0 | 1 | EA = , R + ACCB Offset |
| 1 | R | R | 1 | 0 | 1 | 1 | 0 | EA = , R + ACCA Offset |
| 1 | R | R | 1 | 1 | 0 | 0 | 0 | EA = , R + 8-Bit Offset |
| 1 | R | R | 1 | 1 | 0 | 0 | 1 | EA = , R + 16-Bit Offset |
| 1 | R | R | 1 | 1 | 0 | 1 | 1 | EA = , R + D Offset |
| 1 | x | x | 1 | 1 | 1 | 0 | 0 | EA = , PC + 8-Bit Offset |
| 1 | x | x | 1 | 1 | 1 | 0 | 1 | EA = , PC + 16-Bit Offset |
| 1 | R | R | 1 | 1 | 1 | 1 | 1 | EA = [, Address] |



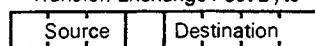
6809 PROGRAMMING MODEL



Push/Pull Post Byte



Transfer/Exchange Post Byte



Register Field

| | |
|----------------|------------|
| 0000 = D (A-B) | 0101 = PC |
| 0001 = X | 1000 = A |
| 0010 = Y | 1001 = B |
| 0011 = U | 1010 = CCR |
| 0100 = S | 1011 = DPR |

6809 Stacking Order

Pull Order

| | |
|--------|---------------|
| CC | |
| A | |
| B | |
| DP | 6809 Vectors |
| X Hi | FFFE Restart |
| X Lo | FFFC NMI |
| Y Hi | FFFA SWI |
| Y Lo | FFF8 IRQ |
| U/S Hi | FFF6 FIRQ |
| U/S Lo | FFF4 SW12 |
| PC Hi | FFF2 SW13 |
| PC Lo | FFF0 Reserved |

Push Order

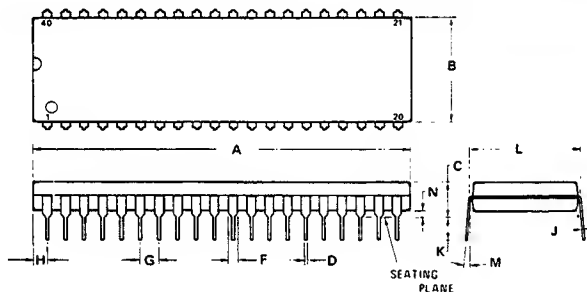
Increasing Memory

ORDERING INFORMATION

| Package Type | Frequency | Temperature Range | Order Number |
|---------------------|-----------|-------------------|--------------|
| Ceramic L Suffix | 1.0 MHz | 0°C to 70°C | MC6809EL |
| | 1.0 MHz | -40°C to 85°C | MC6809ECL |
| | 1.5 MHz | 0°C to 70°C | MC68A09EL |
| | 1.5 MHz | -40°C to 85°C | MC68A09ECL |
| | 2.0 MHz | 0°C to 70°C | MC68B09EL |
| | 2.0 MHz | -40°C to 85°C | MC68B09ECL |
| Plastic P Suffix | 1.0 MHz | 0°C to 70°C | MC6809EP |
| | 1.0 MHz | -40°C to 85°C | MC6809ECP |
| | 1.5 MHz | 0°C to 70°C | MC68A09EP |
| | 1.5 MHz | -40°C to 85°C | MC68A09ECP |
| | 2.0 MHz | 0°C to 70°C | MC68B09EP |
| | 2.0 MHz | -40°C to 85°C | MC68B09ECP |
| Cerdip S Suffix | 1.0 MHz | 0°C to 70°C | MC6809ES |
| | 1.0 MHz | -40°C to 85°C | MC6809ECS |
| | 1.5 MHz | 0°C to 70°C | MC68A09ES |
| | 1.5 MHz | -40°C to 85°C | MC68A09ECS |
| | 2.0 MHz | 0°C to 70°C | MC68B09ES |
| | 2.0 MHz | -40°C to 85°C | MC68B09ECS |



PACKAGE DIMENSIONS

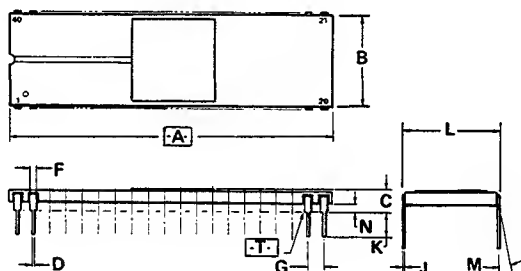


| DIM | MILLIMETERS | | INCHES | |
|-----|-------------|-------|-----------|-------|
| | MIN | MAX | MIN | MAX |
| A | 51.69 | 52.45 | 2.035 | 2.065 |
| B | 13.72 | 14.22 | 0.540 | 0.560 |
| C | 3.94 | 5.08 | 0.155 | 0.200 |
| D | 0.36 | 0.56 | 0.014 | 0.022 |
| F | 1.02 | 1.52 | 0.040 | 0.060 |
| G | 2.54 BSC | | 0.100 BSC | |
| H | 1.65 | 2.16 | 0.065 | 0.085 |
| J | 0.20 | 0.38 | 0.008 | 0.015 |
| K | 2.92 | 3.43 | 0.115 | 0.135 |
| L | 15.24 BSC | | 0.600 BSC | |
| M | 0° | 15° | 0° | 15° |
| N | 0.51 | 1.02 | 0.020 | 0.040 |

NOTES:

1. POSITIONAL TOLERANCE OF LEADS (O), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

P SUFFIX
PLASTIC PACKAGE
CASE 711-03



| DIM | MILLIMETERS | | INCHES | |
|-----|-------------|-------|-----------|-------|
| | MIN | MAX | MIN | MAX |
| A | 50.29 | 51.31 | 1.980 | 2.020 |
| B | 14.63 | 15.49 | 0.576 | 0.610 |
| C | 2.79 | 4.32 | 0.110 | 0.170 |
| D | 0.38 | 0.53 | 0.015 | 0.021 |
| F | 0.76 | 1.52 | 0.030 | 0.060 |
| G | 2.54 BSC | | 0.100 BSC | |
| J | 0.20 | 0.33 | 0.008 | 0.013 |
| K | 2.54 | 4.57 | 0.100 | 0.180 |
| L | 14.99 | 15.65 | 0.590 | 0.616 |
| M | — | 10° | — | 10° |
| N | 1.02 | 1.52 | 0.040 | 0.060 |

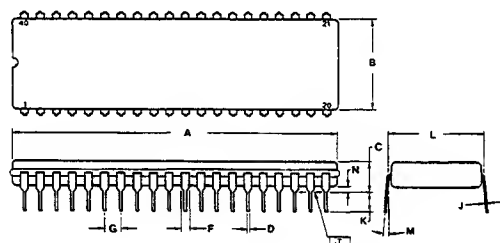
NOTES:

1. DIMENSION [A] IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:

$$\oplus 0.25 (0.010) (M) T A (M)$$

3. [T] IS SEATING PLANE.
4. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973.

L SUFFIX
CERAMIC PACKAGE
CASE 715-05



| DIM | MILLIMETERS | | INCHES | |
|-----|-------------|-------|-----------|-------|
| | MIN | MAX | MIN | MAX |
| A | 51.31 | 53.24 | 2.020 | 2.096 |
| B | 12.70 | 15.49 | 0.500 | 0.610 |
| C | 4.06 | 5.84 | 0.160 | 0.230 |
| D | 0.38 | 0.56 | 0.015 | 0.022 |
| F | 1.27 | 1.65 | 0.050 | 0.065 |
| G | 2.54 BSC | | 0.100 BSC | |
| J | 0.20 | 0.30 | 0.008 | 0.012 |
| K | 3.18 | 4.06 | 0.125 | 0.160 |
| L | 15.24 BSC | | 0.600 BSC | |
| M | 5° | 15° | 5° | 15° |
| N | 0.51 | 1.27 | 0.020 | 0.050 |

NOTES:


1. DIMENSION-A-IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:

$$\oplus 0.25 (0.010) (M) T A (M)$$

3. [T] IS SEATING PLANE.
4. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSION A AND B INCLUDES MENISCUS.

S SUFFIX
CERDIP PACKAGE
CASE 734-03



Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.

ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.



MOTOROLA

A13321-3 PRINTED IN USA (1994) MPS/POO

OS9846-R2

MC6809E/D

